UNIVERSIDADE FEDERAL DE JUIZ DE FORA
Programa de Pós-Graduação em Matemática
Análise Numérica

Luís Karlos Mendes de Oliveira 202069032A

Atividade 02: Métodos Numéricos

Prof.: Sandro Rodrigues Mazorche 14 de outubro de 2025

Exercício 1

Este trabalho tem como objetivo a resolução de Problemas de Valor Inicial (PVIs) por meio da implementação e análise de um conjunto de métodos numéricos de passo simples e de múltiplos passos, avaliando e comparando suas características de exatidão. Os métodos implementados foram o de Euler Explícito (MEE), o da Série de Taylor (MST) de ordens 2, 3, 4 e 6, os de Runge-Kutta (MRK) de ordens 2, 3, 4 e 6, e os de Adams-Bashforth (MMP), também de ordens 2, 3, 4 e 6. As derivadas de ordem superior exigidas pelo MST foram obtidas de forma computacional por meio da biblioteca `SymPy` em Python. As simulações foram realizadas com passos $h = 0.1$ e $h = 0.05$, e as aproximações foram comparadas em pontos específicos do domínio, como $t \in \{0.1, 0.2, 0.3, 0.4\}$.

Para os casos sem solução exata conhecida, utilizou-se como referência uma aproximação, gerada pelo método RK de ordem 6 com passo de simulação $h = 0.001$. A acurácia de cada método foi avaliada por meio do erro local e do erro relativo em relação à solução exata ou de referência. Por fim, os resultados numéricos foram apresentados e discutidos, comparando a performance dos diferentes métodos.

Exercício 1 - letra A

O Problema de Valor Inicial (PVI) a ser resolvido é dado por:

$$\begin{cases} x'(t) = 3 + t - x(t) \\ x(0) = 1 \end{cases} \tag{1}$$

Solução Analítica

A EDO linear de primeira ordem $x'(t) + x(t) = 3 + t$ é resolvida utilizando o fator integrante $I(t) = e^t$, como abaixo:

$$\begin{aligned} \frac{d}{dt}\left(e^t x(t)\right) &= 3e^t + te^t \\ e^t x(t) &= \int (3e^t + te^t)\, dt \\ e^t x(t) &= 3e^t + (te^t - e^t) + C \\ x(t) &= 2 + t + Ce^{-t} \end{aligned}$$

A constante C é determinada pela condição inicial $x(0) = 1$:

$$1 = 2 + 0 + C \implies C = -1$$

Portanto, a solução exata para o PVI é:

$$x(t) = 2 + t - e^{-t} \tag{2}$$

Método de Euler Explícito (MEE)

Nas tabelas a seguir e no gráfico 1 são apresentados os valores aproximados do PVI (1) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Euler Explícito (MEE) com passo $h = 0.1$ e $h = 0.05$ comparados com a solução analítica.

Tabela 1: Resultados da simulação MEE para h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.200000000	0.004837418	0.404749790
0.200000000	1.381269247	1.390000000	0.008730753	0.632081913
0.300000000	1.559181779	1.571000000	0.011818221	0.757975807
0.400000000	1.729679954	1.743900000	0.014220046	0.822120069

Tabela 2: Resultados da simulação MEE para h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.197500000	0.002337418	0.195573228
0.200000000	1.381269247	1.385493750	0.004224503	0.305842115
0.300000000	1.559181779	1.564908109	0.005726330	0.367265070
0.400000000	1.729679954	1.736579569	0.006899615	0.398895456

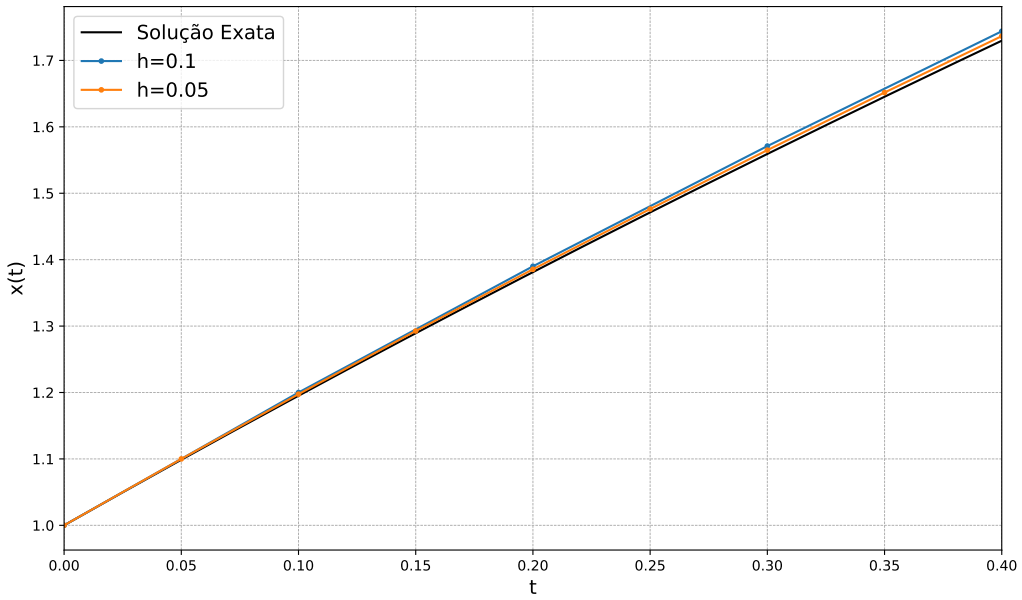

Figura 1: Comparação entre a solução exata e as aproximações obtidas pelo MEE

Método da Série de Taylor (MST)

Nas tabelas a seguir e no gráfico 2 são apresentados os valores aproximados do PVI (1) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método da Série de Taylor (MST) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução analítica.

Tabela 3: Resultados da Série de Taylor ordem 2 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195000000	0.000162582	0.013603335
0.200000000	1.381269247	1.380975000	0.000294247	0.021302648
0.300000000	1.559181779	1.558782375	0.000399404	0.025616277
0.400000000	1.729679954	1.729198049	0.000481905	0.027860911

Tabela 4: Resultados da Série de Taylor ordem 3 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195166667	0.000004085	0.000341770
0.200000000	1.381269247	1.381276639	0.000007392	0.000535158
0.300000000	1.559181779	1.559191812	0.000010033	0.000643464
0.400000000	1.729679954	1.729692058	0.000012104	0.000699783

Tabela 5: Resultados da Série de Taylor ordem 4 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162500	0.000000082	0.000006858
0.200000000	1.381269247	1.381269099	0.000000148	0.000010739
0.300000000	1.559181779	1.559181578	0.000000201	0.000012912
0.400000000	1.729679954	1.729679711	0.000000243	0.000014042

Tabela 6: Resultados da Série de Taylor ordem 6 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162582	0.000000000	0.000000002
0.200000000	1.381269247	1.381269247	0.000000000	0.000000003
0.300000000	1.559181779	1.559181779	0.000000000	0.000000003
0.400000000	1.729679954	1.729679954	0.000000000	0.000000003

Tabela 7: Resultados da Série de Taylor ordem 2 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195123437	0.000039144	0.003275242
0.200000000	1.381269247	1.381198407	0.000070840	0.005128637
0.300000000	1.559181779	1.559085629	0.000096150	0.006166727
0.400000000	1.729679954	1.729563951	0.000116003	0.006706631

Tabela 8: Resultados da Série de Taylor ordem 3 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195163072	0.000000491	0.000041042
0.200000000	1.381269247	1.381270135	0.000000888	0.000064265
0.300000000	1.559181779	1.559182984	0.000001205	0.000077272
0.400000000	1.729679954	1.729681408	0.000001454	0.000084035

Tabela 9: Resultados da Série de Taylor ordem 4 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162577	0.000000005	0.000000411
0.200000000	1.381269247	1.381269238	0.000000009	0.000000644
0.300000000	1.559181779	1.559181767	0.000000012	0.000000774
0.400000000	1.729679954	1.729679939	0.000000015	0.000000842

Tabela 10: Resultados da Série de Taylor ordem 6 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162582	0.000000000	0.000000000
0.200000000	1.381269247	1.381269247	0.000000000	0.000000000
0.300000000	1.559181779	1.559181779	0.000000000	0.000000000
0.400000000	1.729679954	1.729679954	0.000000000	0.000000000

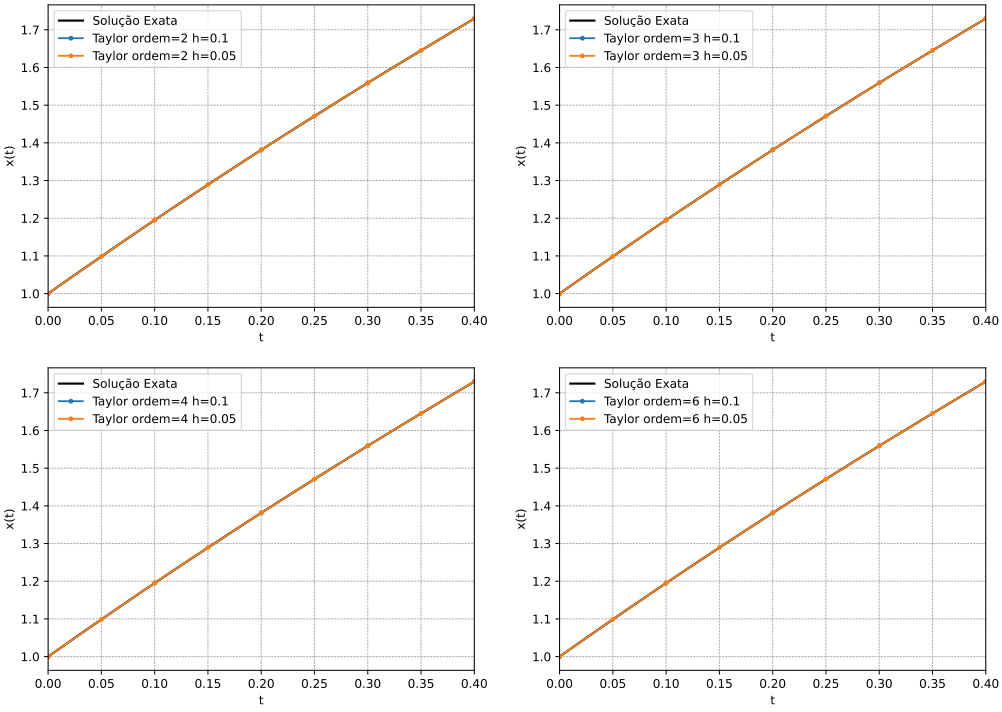

Figura 2: Comparação entre a solução exata e as aproximações obtidas pelo MST

Método de Runge-Kutta (MRK)

Nas tabelas a seguir e no gráfico 3 são apresentados os valores aproximados do PVI (1) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Runge-Kutta (MRK) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução analítica.

Tabela 11: Resultados da simulação RK para RK2 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195000000	0.000162582	0.013603335
0.200000000	1.381269247	1.380975000	0.000294247	0.021302648
0.300000000	1.559181779	1.558782375	0.000399404	0.025616277
0.400000000	1.729679954	1.729198049	0.000481905	0.027860911

Tabela 12: Resultados da simulação RK para RK3 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195166667	0.000004085	0.000341770
0.200000000	1.381269247	1.381276639	0.000007392	0.000535158
0.300000000	1.559181779	1.559191812	0.000010033	0.000643464
0.400000000	1.729679954	1.729692058	0.000012104	0.000699783

Tabela 13: Resultados da simulação RK para RK4 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162500	0.000000082	0.000006858
0.200000000	1.381269247	1.381269099	0.000000148	0.000010739
0.300000000	1.559181779	1.559181578	0.000000201	0.000012912
0.400000000	1.729679954	1.729679711	0.000000243	0.000014042

Tabela 14: Resultados da simulação RK para RK6 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162582	0.000000000	0.000000006
0.200000000	1.381269247	1.381269247	0.000000000	0.000000009
0.300000000	1.559181779	1.559181779	0.000000000	0.000000010
0.400000000	1.729679954	1.729679954	0.000000000	0.000000011

Tabela 15: Resultados da simulação RK para RK2 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195123437	0.000039144	0.003275242
0.200000000	1.381269247	1.381198407	0.000070840	0.005128637
0.300000000	1.559181779	1.559085629	0.000096150	0.006166727
0.400000000	1.729679954	1.729563951	0.000116003	0.006706631

Tabela 16: Resultados da simulação RK para RK3 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195163072	0.000000491	0.000041042
0.200000000	1.381269247	1.381270135	0.000000888	0.000064265
0.300000000	1.559181779	1.559182984	0.000001205	0.000077272
0.400000000	1.729679954	1.729681408	0.000001454	0.000084035

Tabela 17: Resultados da simulação RK para RK4 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162577	0.000000005	0.000000411
0.200000000	1.381269247	1.381269238	0.000000009	0.000000644
0.300000000	1.559181779	1.559181767	0.000000012	0.000000774
0.400000000	1.729679954	1.729679939	0.000000015	0.000000842

Tabela 18: Resultados da simulação RK para RK6 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162582	0.000000000	0.000000000
0.200000000	1.381269247	1.381269247	0.000000000	0.000000000
0.300000000	1.559181779	1.559181779	0.000000000	0.000000000
0.400000000	1.729679954	1.729679954	0.000000000	0.000000000

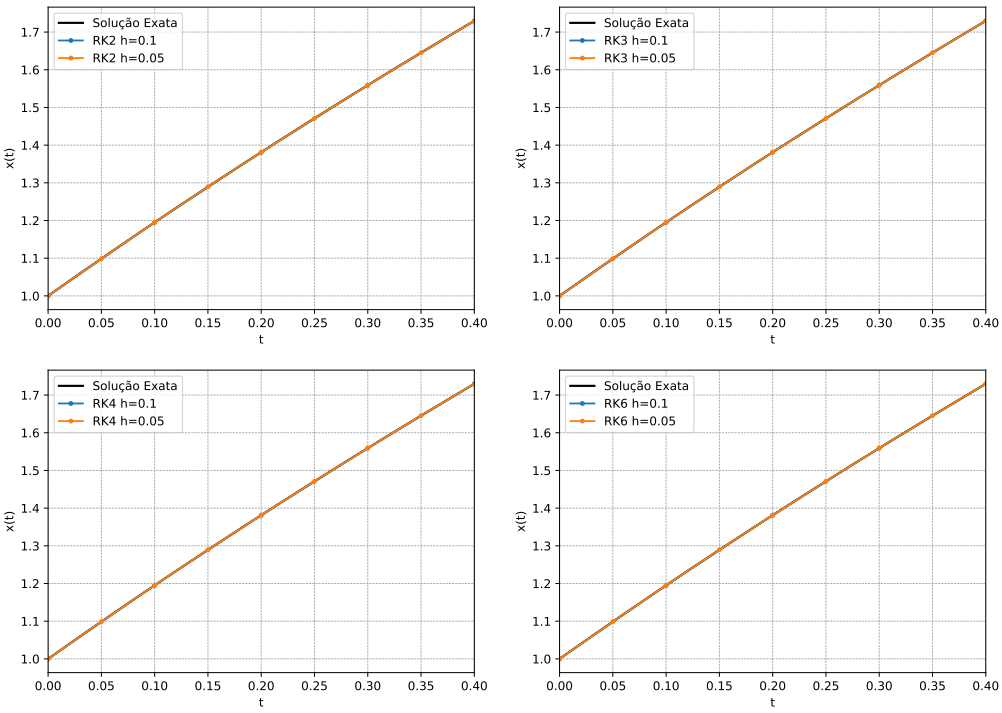

Figura 3: Comparação entre a solução exata e as aproximações obtidas pelo MRK

Método de Multi-Passo (MMP)

Nas tabelas a seguir e no gráfico 4 são apresentados os valores aproximados do PVI (1) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Multi-Passo (MMP) com passo

$h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução analítica. Foi utilizado o método RK6 para obter os passos iniciais necessários para o MMP.

Tabela 19: Resultados da simulação AB para AB2 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162500	0.000000082	0.000006858
0.200000000	1.381269247	1.380888125	0.000381122	0.027592153
0.300000000	1.559181779	1.558513031	0.000668748	0.042890962
0.400000000	1.729679954	1.728780483	0.000899471	0.052002172

Tabela 20: Resultados da simulação AB para AB3 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162582	0.000000000	0.000000006
0.200000000	1.381269247	1.381269247	0.000000000	0.000000009
0.300000000	1.559181779	1.559214319	0.000032539	0.002086956
0.400000000	1.729679954	1.729735700	0.000055746	0.003222890

Tabela 21: Resultados da simulação AB para AB4 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162582	0.000000000	0.000000006
0.200000000	1.381269247	1.381269247	0.000000000	0.000000009
0.300000000	1.559181779	1.559181779	0.000000000	0.000000010
0.400000000	1.729679954	1.729677080	0.000002874	0.000166162

Tabela 22: Resultados da simulação AB para AB6 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162582	0.000000000	0.000000006
0.200000000	1.381269247	1.381269247	0.000000000	0.000000009
0.300000000	1.559181779	1.559181779	0.000000000	0.000000010
0.400000000	1.729679954	1.729679954	0.000000000	0.000000011

Tabela 23: Resultados da simulação AB para AB2 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195112780	0.000049802	0.004166966
0.200000000	1.381269247	1.381136511	0.000132736	0.009609682
0.300000000	1.559181779	1.558982308	0.000199472	0.012793359
0.400000000	1.729679954	1.729427642	0.000252312	0.014587205

Tabela 24: Resultados da simulação AB para AB3 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162582	0.000000000	0.000000000
0.200000000	1.381269247	1.381273297	0.000004050	0.000293185
0.300000000	1.559181779	1.559189110	0.000007331	0.000470186
0.400000000	1.729679954	1.729689903	0.000009949	0.000575216

Tabela 25: Resultados da simulação AB para AB4 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162582	0.000000000	0.000000000
0.200000000	1.381269247	1.381269148	0.000000099	0.000007158
0.300000000	1.559181779	1.559181517	0.000000262	0.000016830
0.400000000	1.729679954	1.729679559	0.000000395	0.000022846

Tabela 26: Resultados da simulação AB para AB6 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.195162582	1.195162582	0.000000000	0.000000000
0.200000000	1.381269247	1.381269247	0.000000000	0.000000000
0.300000000	1.559181779	1.559181779	0.000000000	0.000000014
0.400000000	1.729679954	1.729679953	0.000000001	0.000000033

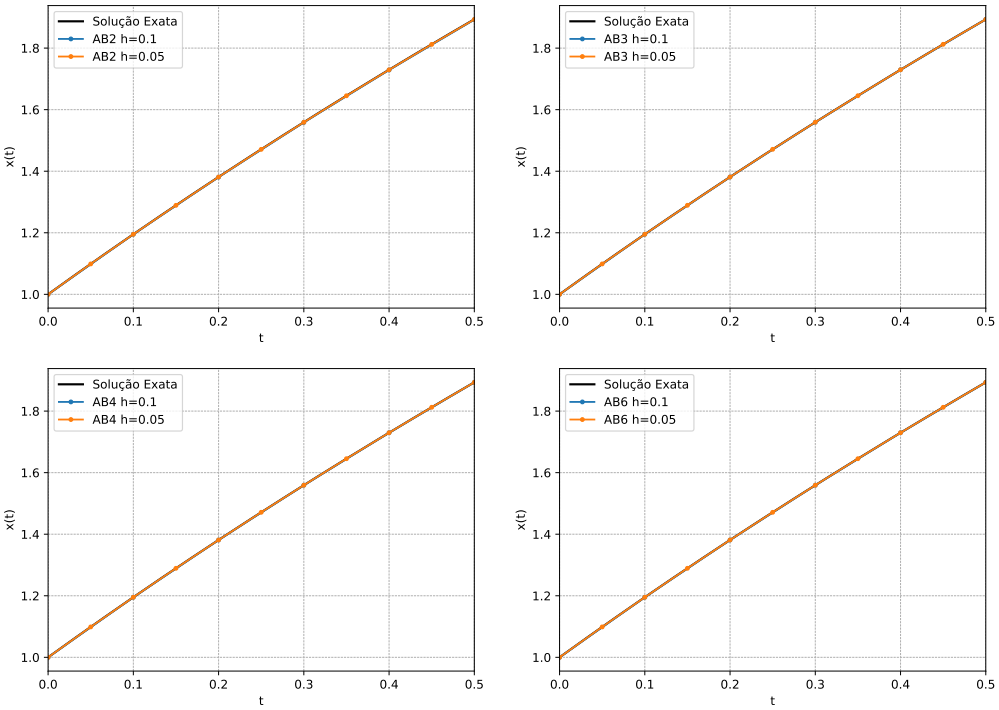

Figura 4: Comparação entre a solução exata e as aproximações obtidas pelo MMP

Exercício 1 - letra B

O Problema de Valor Inicial (PVI) a ser resolvido é dado por:

$$\begin{cases} x'(t) = 5t - 3\sqrt{x(t)} \\ x(0) = 2 \end{cases} \tag{3}$$

Solução Analítica

A Equação Diferencial Ordinária (EDO) $x'(t) = 5t - 3\sqrt{x(t)}$ é não linear e não possui uma solução analítica conhecida em termos de funções elementares. Portanto, a análise numérica será realizada comparando os resultados com uma solução de referência, gerada pelo método de Runge-Kutta de 6ª ordem (RK6) com um passo de simulação $h = 0.001$.

Método de Euler Explícito (MEE)

Nas tabelas a seguir e no gráfico 5 são apresentados os valores aproximados do PVI (3) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Euler Explícito (MEE) com passo $h = 0.1$ e $h = 0.05$ comparados com a solução aproximada (RK6 com $h = 0.001$).

Tabela 27: Resultados da simulação MEE para h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.575735931	0.046568038	2.870487843
0.200000000	1.333617720	1.249150969	0.084466751	6.333655387
0.300000000	1.126848934	1.013854701	0.112994233	10.027451745
0.400000000	0.993825483	0.861783645	0.132041838	13.286219831

Tabela 28: Resultados da simulação MEE para h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.599801196	0.022502774	1.387087395
0.200000000	1.333617720	1.292884317	0.040733402	3.054353706
0.300000000	1.126848934	1.072415655	0.054433279	4.830574668
0.400000000	0.993825483	0.930174626	0.063650857	6.404631252

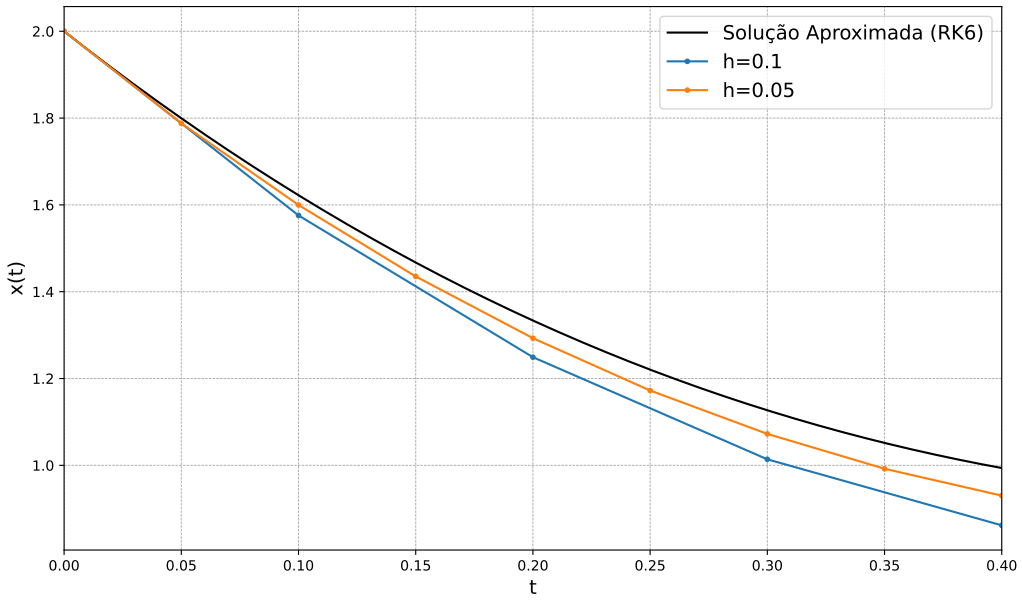

Figura 5: Comparação entre a solução aproximada (RK6 com $h = 0.001$) e as aproximações obtidas pelo MEE

Método da Série de Taylor (MST)

Nas tabelas a seguir e no gráfico 6 são apresentados os valores aproximados do PVI (3) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método da Série de Taylor (MST) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução aproximada (RK6 com $h = 0.001$).

Tabela 29: Resultados da Série de Taylor ordem 2 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.623235931	0.000931962	0.057446804
0.200000000	1.333617720	1.335573761	0.001956042	0.146671860
0.300000000	1.126848934	1.129882942	0.003034007	0.269247026
0.400000000	0.993825483	0.997911428	0.004085944	0.411132976

Tabela 30: Resultados da Série de Taylor ordem 3 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622352048	0.000048078	0.002963582
0.200000000	1.333617720	1.333711323	0.000093603	0.007018727
0.300000000	1.126848934	1.126975399	0.000126465	0.011222855
0.400000000	0.993825483	0.993959286	0.000133802	0.013463380

Tabela 31: Resultados da Série de Taylor ordem 4 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622305173	0.000001203	0.000074172
0.200000000	1.333617720	1.333618709	0.000000989	0.000074183
0.300000000	1.126848934	1.126847313	0.000001621	0.000143852
0.400000000	0.993825483	0.993818722	0.000006761	0.000680343

Tabela 32: Resultados da Série de Taylor ordem 6 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622303949	0.000000021	0.000001292
0.200000000	1.333617720	1.333617672	0.000000047	0.000003557
0.300000000	1.126848934	1.126848877	0.000000057	0.000005056
0.400000000	0.993825483	0.993825465	0.000000018	0.000001795

Tabela 33: Resultados da Série de Taylor ordem 2 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622536806	0.000232837	0.014352228
0.200000000	1.333617720	1.334104477	0.000486757	0.036498992
0.300000000	1.126848934	1.127599855	0.000750921	0.066639018
0.400000000	0.993825483	0.994829891	0.001004408	0.101064824

Tabela 34: Resultados da Série de Taylor ordem 3 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622309906	0.000005936	0.000365927
0.200000000	1.333617720	1.333629105	0.000011385	0.000853690
0.300000000	1.126848934	1.126863973	0.000015038	0.001334543
0.400000000	0.993825483	0.993840864	0.000015380	0.001547591

Tabela 35: Resultados da Série de Taylor ordem 4 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622304032	0.000000063	0.000003875
0.200000000	1.333617720	1.333617748	0.000000029	0.000002147
0.300000000	1.126848934	1.126848778	0.000000157	0.000013889
0.400000000	0.993825483	0.993825002	0.000000481	0.000048428

Tabela 36: Resultados da Série de Taylor ordem 6 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622303969	0.000000000	0.000000022
0.200000000	1.333617720	1.333617719	0.000000001	0.000000055
0.300000000	1.126848934	1.126848934	0.000000001	0.000000067
0.400000000	0.993825483	0.993825483	0.000000000	0.000000002

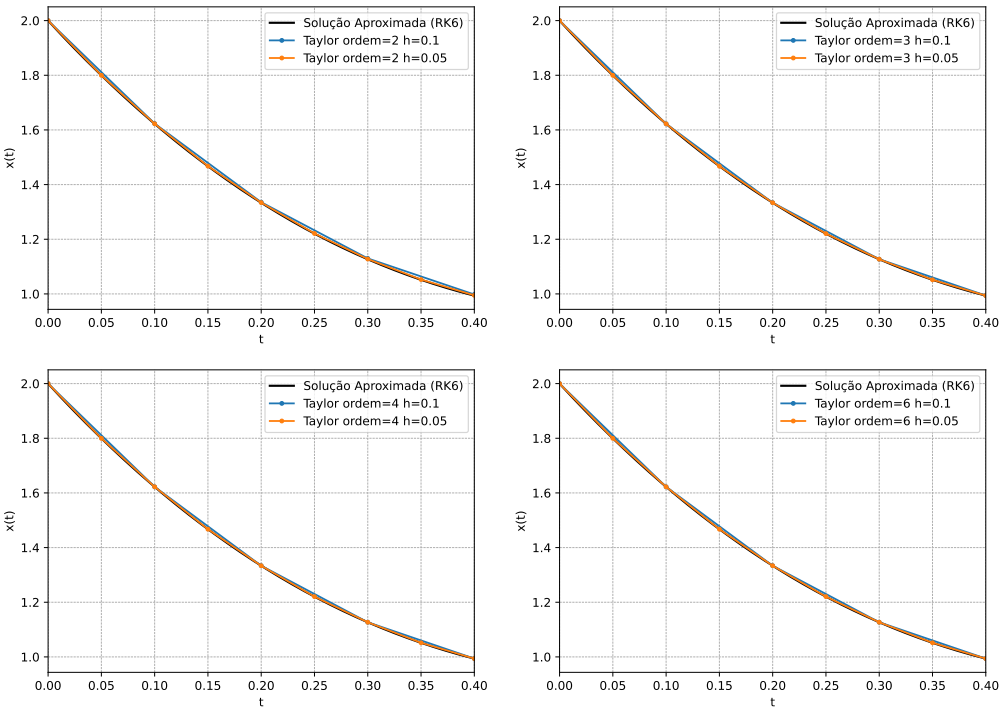

Figura 6: Comparação entre a solução aproximada (RK6 com $h = 0.001$) e as aproximações obtidas pelo MST

Método de Runge-Kutta (MRK)

Nas tabelas a seguir e no gráfico 7 são apresentados os valores aproximados do PVI (3) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Runge-Kutta (MRK) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução aproximada (RK6 com $h = 0.001$).

Tabela 37: Resultados da simulação RK para RK2 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.624575485	0.002271515	0.140017841
0.200000000	1.333617720	1.337874835	0.004257116	0.319215596
0.300000000	1.126848934	1.132707568	0.005858634	0.519912975
0.400000000	0.993825483	1.000833420	0.007007937	0.705147637

Tabela 38: Resultados da simulação RK para RK3 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622214101	0.000089869	0.005539585
0.200000000	1.333617720	1.333444989	0.000172731	0.012952040
0.300000000	1.126848934	1.126607605	0.000241329	0.021416284
0.400000000	0.993825483	0.993536543	0.000288940	0.029073535

Tabela 39: Resultados da simulação RK para RK4 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622307372	0.000003402	0.000209720
0.200000000	1.333617720	1.333624700	0.000006981	0.000523434
0.300000000	1.126848934	1.126859401	0.000010467	0.000928888
0.400000000	0.993825483	0.993838947	0.000013464	0.001354753

Tabela 40: Resultados da simulação RK para RK6 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622303970	0.000000000	0.000000007
0.200000000	1.333617720	1.333617719	0.000000000	0.000000019
0.300000000	1.126848934	1.126848933	0.000000001	0.000000115
0.400000000	0.993825483	0.993825480	0.000000003	0.000000286

Tabela 41: Resultados da simulação RK para RK2 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622831708	0.000527738	0.032530184
0.200000000	1.333617720	1.334604231	0.000986512	0.073972597
0.300000000	1.126848934	1.128203984	0.001355050	0.120251268
0.400000000	0.993825483	0.995445045	0.001619561	0.162962341

Tabela 42: Resultados da simulação RK para RK3 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622292802	0.000011167	0.000688371
0.200000000	1.333617720	1.333596195	0.000021525	0.001614022
0.300000000	1.126848934	1.126818746	0.000030188	0.002678959
0.400000000	0.993825483	0.993789172	0.000036312	0.003653729

Tabela 43: Resultados da simulação RK para RK4 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622304171	0.000000201	0.000012401
0.200000000	1.333617720	1.333618131	0.000000412	0.000030863
0.300000000	1.126848934	1.126849550	0.000000616	0.000054633
0.400000000	0.993825483	0.993826274	0.000000791	0.000079545

Tabela 44: Resultados da simulação RK para RK6 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622303970	0.000000000	0.000000000
0.200000000	1.333617720	1.333617720	0.000000000	0.000000001
0.300000000	1.126848934	1.126848934	0.000000000	0.000000002
0.400000000	0.993825483	0.993825483	0.000000000	0.000000005

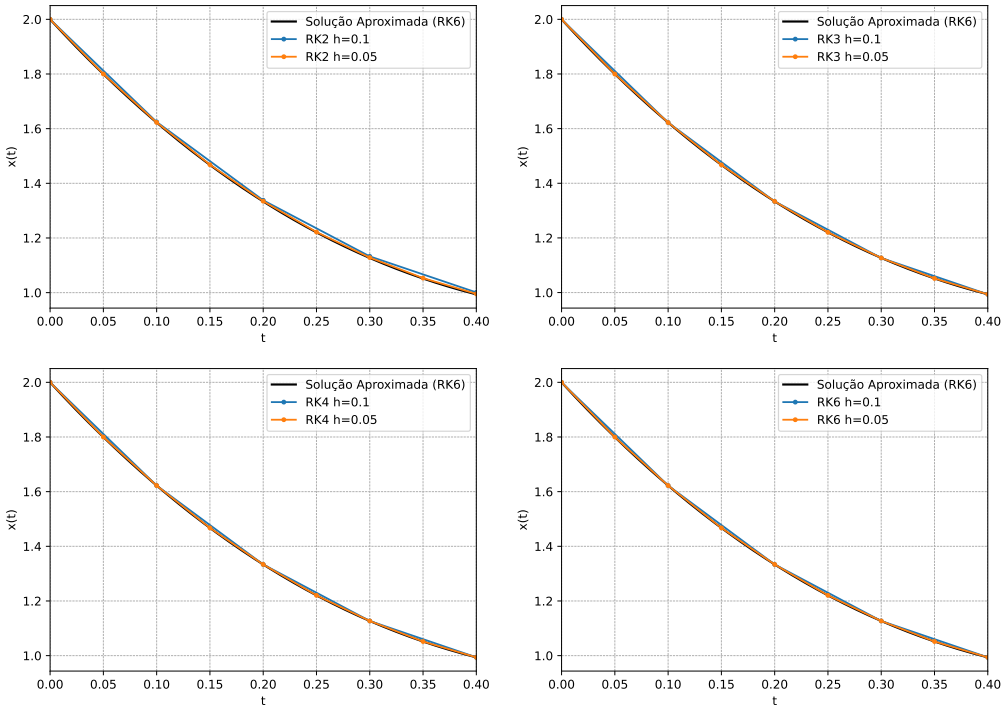

Figura 7: Comparação entre a solução aproximada (RK6 com $h = 0.001$) e as aproximações obtidas pelo MRK

Método de Multi-Passo (MMP)

Nas tabelas a seguir e no gráfico 8 são apresentados os valores aproximados do PVI (3) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Multi-Passo (MMP) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução aproximada (RK6 com $h = 0.001$). Foi utilizado o método RK6 para obter os passos iniciais necessários para o MMP.

Tabela 45: Resultados da simulação AB para AB2 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622307372	0.000003402	0.000209720
0.200000000	1.333617720	1.336275169	0.002657449	0.199266206
0.300000000	1.126848934	1.132141755	0.005292820	0.469700978
0.400000000	0.993825483	1.001728133	0.007902649	0.795174755

Tabela 46: Resultados da simulação AB para AB3 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622303970	0.000000000	0.000000007
0.200000000	1.333617720	1.333617719	0.000000000	0.000000019
0.300000000	1.126848934	1.127296199	0.000447265	0.039691614
0.400000000	0.993825483	0.994511864	0.000686381	0.069064501

Tabela 47: Resultados da simulação AB para AB4 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622303970	0.000000000	0.000000007
0.200000000	1.333617720	1.333617719	0.000000000	0.000000019
0.300000000	1.126848934	1.126848933	0.000000001	0.000000115
0.400000000	0.993825483	0.993742727	0.000082757	0.008327086

Tabela 48: Resultados da simulação AB para AB6 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622303970	0.000000000	0.000000007
0.200000000	1.333617720	1.333617719	0.000000000	0.000000019
0.300000000	1.126848934	1.126848933	0.000000001	0.000000115
0.400000000	0.993825483	0.993825480	0.000000003	0.000000286

Tabela 49: Resultados da simulação AB para AB2 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622607510	0.000303540	0.018710433
0.200000000	1.333617720	1.334548287	0.000930568	0.069777686
0.300000000	1.126848934	1.128433150	0.001584216	0.140588131
0.400000000	0.993825483	0.996042768	0.002217285	0.223106057

Tabela 50: Resultados da simulação AB para AB3 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622303970	0.000000000	0.000000000
0.200000000	1.333617720	1.333671378	0.000053659	0.004023557
0.300000000	1.126848934	1.126944781	0.000095847	0.008505754
0.400000000	0.993825483	0.993939189	0.000113706	0.011441228

Tabela 51: Resultados da simulação AB para AB4 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622303970	0.000000000	0.000000000
0.200000000	1.333617720	1.333618019	0.000000299	0.000022437
0.300000000	1.126848934	1.126845822	0.000003112	0.000276200
0.400000000	0.993825483	0.993812742	0.000012741	0.001282042

Tabela 52: Resultados da simulação AB para AB6 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.622303970	1.622303970	0.000000000	0.000000000
0.200000000	1.333617720	1.333617720	0.000000000	0.000000001
0.300000000	1.126848934	1.126848609	0.000000326	0.000028914
0.400000000	0.993825483	0.993824979	0.000000504	0.000050743

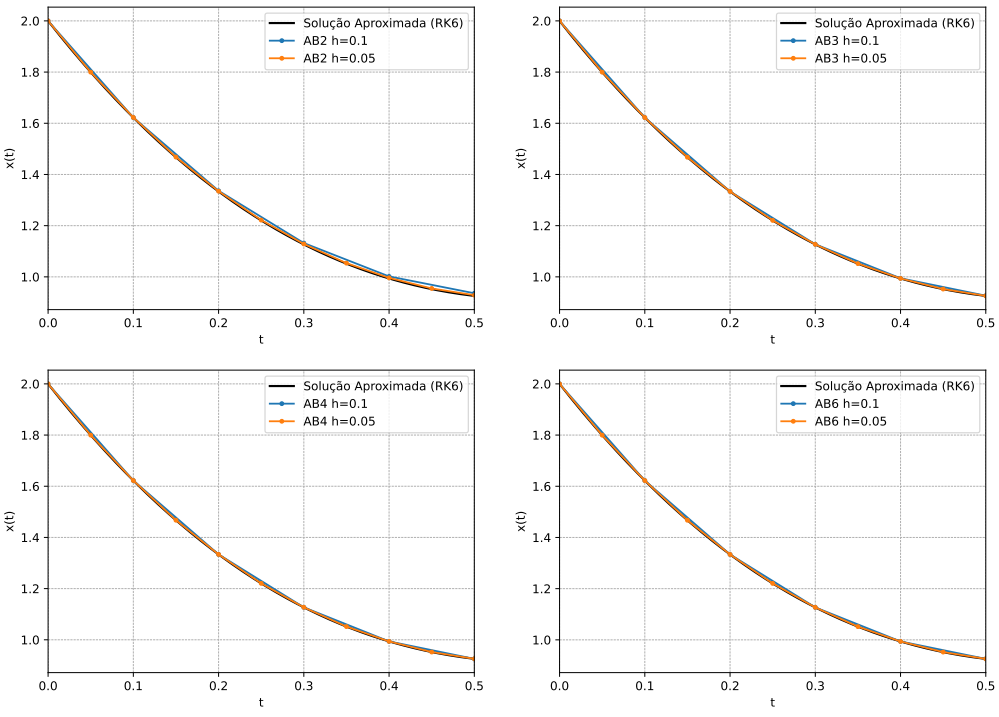

Figura 8: Comparação entre a solução aproximada (RK6 com $h = 0.001$) e as aproximações obtidas pelo MMP

Exercício 1 - letra C

O Problema de Valor Inicial (PVI) a ser resolvido é dado por:

$$\begin{cases} x'(t) = 2x(t) - 3t \\ x(0) = 1 \end{cases} \tag{4}$$

Solução Analítica

A Equação Diferencial Ordinária (EDO) $x'(t) - 2x(t) = -3t$ é linear de primeira ordem e resolvida utilizando o fator integrante $I(t) = e^{-2t}$, como a seguir:

$$\begin{aligned} \frac{d}{dt}\left(e^{-2t}x(t)\right) &= -3te^{-2t} \\ e^{-2t}x(t) &= \int -3te^{-2t}\,dt \\ e^{-2t}x(t) &= \frac{3}{2}te^{-2t} + \frac{3}{4}e^{-2t} + C \\ x(t) &= \frac{3}{2}t + \frac{3}{4} + Ce^{2t} \end{aligned}$$

A constante C é determinada pela condição inicial $x(0) = 1$:

$$1 = \frac{3}{2}(0) + \frac{3}{4} + C \cdot e^0 \implies C = 1 - \frac{3}{4} = \frac{1}{4}$$

Portanto, a solução exata para o PVI é:

$$x(t) = \frac{3}{2}t + \frac{3}{4} + \frac{1}{4}e^{2t} \tag{5}$$

Método de Euler Explícito (MEE)

Nas tabelas a seguir e no gráfico 9 são apresentados os valores aproximados do PVI (4) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Euler Explícito (MEE) com passo $h = 0.1$ e $h = 0.05$ comparados com a solução analítica.

Tabela 53: Resultados da simulação MEE para h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.200000000	0.005350690	0.443911435
0.200000000	1.422956174	1.410000000	0.012956174	0.910511135
0.300000000	1.655529700	1.632000000	0.023529700	1.421279250
0.400000000	1.906385232	1.868400000	0.037985232	1.992526562

Tabela 54: Resultados da simulação MEE para h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.202500000	0.002850690	0.236502917
0.200000000	1.422956174	1.416025000	0.006931174	0.487096829
0.300000000	1.655529700	1.642890250	0.012639450	0.763468641
0.400000000	1.906385232	1.885897203	0.020488030	1.074705641

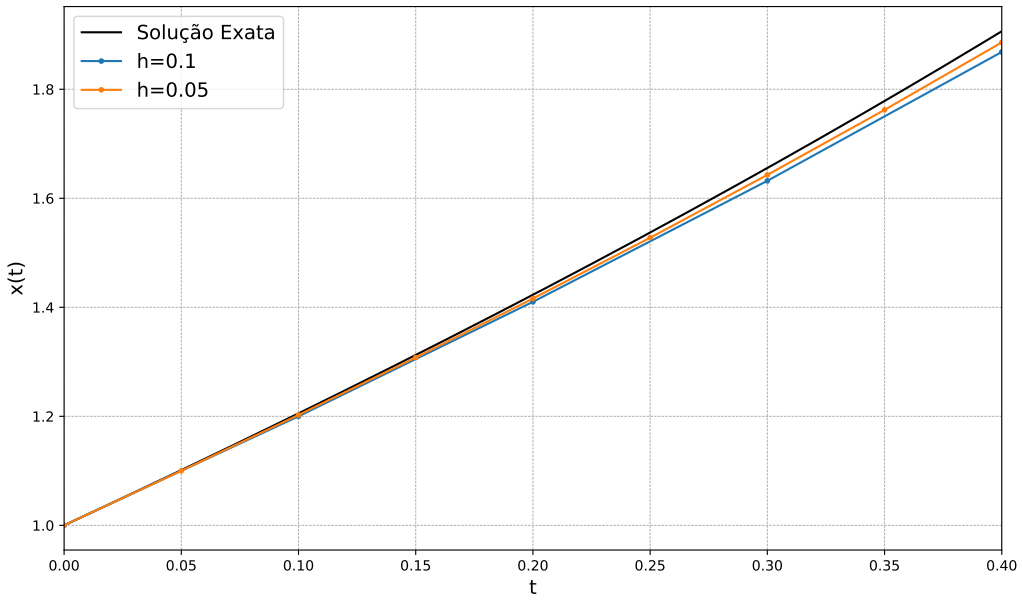

Figura 9: Comparação entre a solução exata e as aproximações obtidas pelo MEE

Método da Série de Taylor (MST)

Nas tabelas a seguir e no gráfico 10 são apresentados os valores aproximados do PVI (4) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método da Série de Taylor (MST) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução analítica.

Tabela 55: Resultados da Série de Taylor ordem 2 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205000000	0.000350690	0.029094399
0.200000000	1.422956174	1.422100000	0.000856174	0.060168713
0.300000000	1.655529700	1.653962000	0.001567700	0.094694773
0.400000000	1.906385232	1.903833640	0.002551592	0.133844518

Tabela 56: Resultados da Série de Taylor ordem 3 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205333333	0.000017356	0.001439930
0.200000000	1.422956174	1.422913778	0.000042397	0.002979476
0.300000000	1.655529700	1.655452027	0.000077673	0.004691721
0.400000000	1.906385232	1.906258743	0.000126489	0.006635044

Tabela 57: Resultados da Série de Taylor ordem 4 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350000	0.000000690	0.000057207
0.200000000	1.422956174	1.422954490	0.000001684	0.000118374
0.300000000	1.655529700	1.655526614	0.000003086	0.000186406
0.400000000	1.906385232	1.906380206	0.000005026	0.000263623

Tabela 58: Resultados da Série de Taylor ordem 6 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350689	0.000000001	0.000000054
0.200000000	1.422956174	1.422956173	0.000000002	0.000000112
0.300000000	1.655529700	1.655529697	0.000000003	0.000000176
0.400000000	1.906385232	1.906385227	0.000000005	0.000000249

Tabela 59: Resultados da Série de Taylor ordem 2 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205256250	0.000094440	0.007835026
0.200000000	1.422956174	1.422725513	0.000230662	0.016210039
0.300000000	1.655529700	1.655107169	0.000422531	0.025522406
0.400000000	1.906385232	1.905697231	0.000688001	0.036089295

Tabela 60: Resultados da Série de Taylor ordem 3 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205348340	0.000002349	0.000194903
0.200000000	1.422956174	1.422950436	0.000005739	0.000403299
0.300000000	1.655529700	1.655519186	0.000010514	0.000635083
0.400000000	1.906385232	1.906368110	0.000017122	0.000898158

Tabela 61: Resultados da Série de Taylor ordem 4 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350643	0.000000047	0.000003885
0.200000000	1.422956174	1.422956060	0.000000114	0.000008039
0.300000000	1.655529700	1.655529491	0.000000210	0.000012659
0.400000000	1.906385232	1.906384891	0.000000341	0.000017903

Tabela 62: Resultados da Série de Taylor ordem 6 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350690	0.000000000	0.000000001
0.200000000	1.422956174	1.422956174	0.000000000	0.000000002
0.300000000	1.655529700	1.655529700	0.000000000	0.000000003
0.400000000	1.906385232	1.906385232	0.000000000	0.000000004

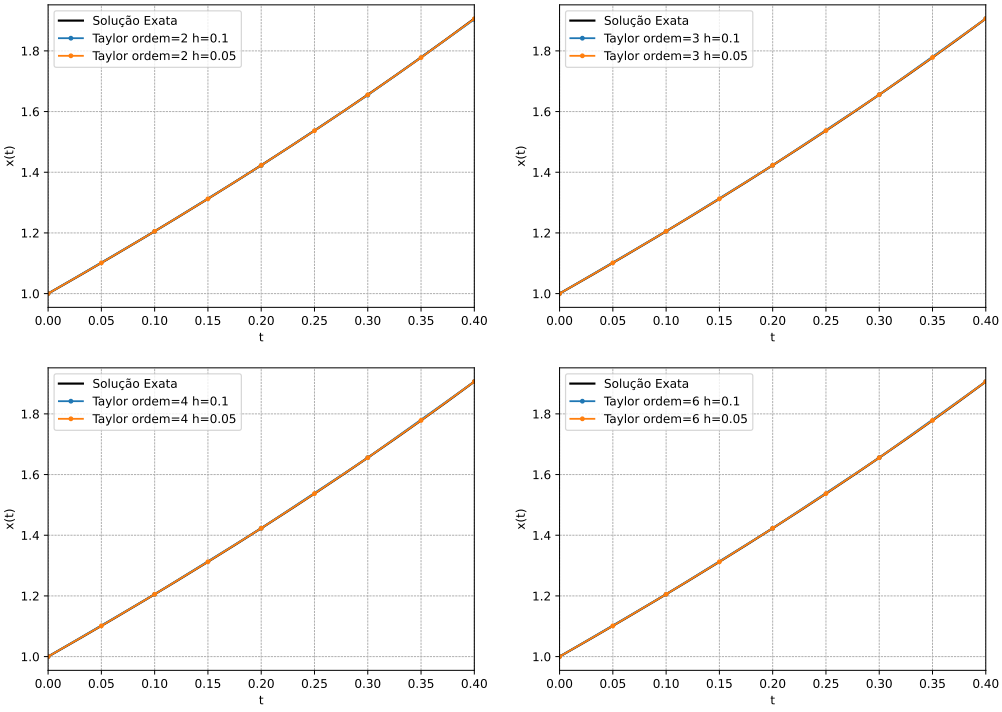

Figura 10: Comparação entre a solução exata e as aproximações obtidas pelo MST

Método de Runge-Kutta (MRK)

Nas tabelas a seguir e no gráfico 11 são apresentados os valores aproximados do PVI (4) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Runge-Kutta (MRK) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução analítica.

Tabela 63: Resultados da simulação RK para RK2 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205000000	0.000350690	0.029094399
0.200000000	1.422956174	1.422100000	0.000856174	0.060168713
0.300000000	1.655529700	1.653962000	0.001567700	0.094694773
0.400000000	1.906385232	1.903833640	0.002551592	0.133844518

Tabela 64: Resultados da simulação RK para RK3 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205333333	0.000017356	0.001439930
0.200000000	1.422956174	1.422913778	0.000042397	0.002979476
0.300000000	1.655529700	1.655452027	0.000077673	0.004691721
0.400000000	1.906385232	1.906258743	0.000126489	0.006635044

Tabela 65: Resultados da simulação RK para RK4 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350000	0.000000690	0.000057207
0.200000000	1.422956174	1.422954490	0.000001684	0.000118374
0.300000000	1.655529700	1.655526614	0.000003086	0.000186406
0.400000000	1.906385232	1.906380206	0.000005026	0.000263623

Tabela 66: Resultados da simulação RK para RK6 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350687	0.000000002	0.000000177
0.200000000	1.422956174	1.422956169	0.000000005	0.000000366
0.300000000	1.655529700	1.655529691	0.000000010	0.000000577
0.400000000	1.906385232	1.906385217	0.000000016	0.000000815

Tabela 67: Resultados da simulação RK para RK2 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205256250	0.000094440	0.007835026
0.200000000	1.422956174	1.422725513	0.000230662	0.016210039
0.300000000	1.655529700	1.655107169	0.000422531	0.025522406
0.400000000	1.906385232	1.905697231	0.000688001	0.036089295

Tabela 68: Resultados da simulação RK para RK3 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205348340	0.000002349	0.000194903
0.200000000	1.422956174	1.422950436	0.000005739	0.000403299
0.300000000	1.655529700	1.655519186	0.000010514	0.000635083
0.400000000	1.906385232	1.906368110	0.000017122	0.000898158

Tabela 69: Resultados da simulação RK para RK4 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350643	0.000000047	0.000003885
0.200000000	1.422956174	1.422956060	0.000000114	0.000008039
0.300000000	1.655529700	1.655529491	0.000000210	0.000012659
0.400000000	1.906385232	1.906384891	0.000000341	0.000017903

Tabela 70: Resultados da simulação RK para RK6 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350690	0.000000000	0.000000003
0.200000000	1.422956174	1.422956174	0.000000000	0.000000006
0.300000000	1.655529700	1.655529700	0.000000000	0.000000010
0.400000000	1.906385232	1.906385232	0.000000000	0.000000014

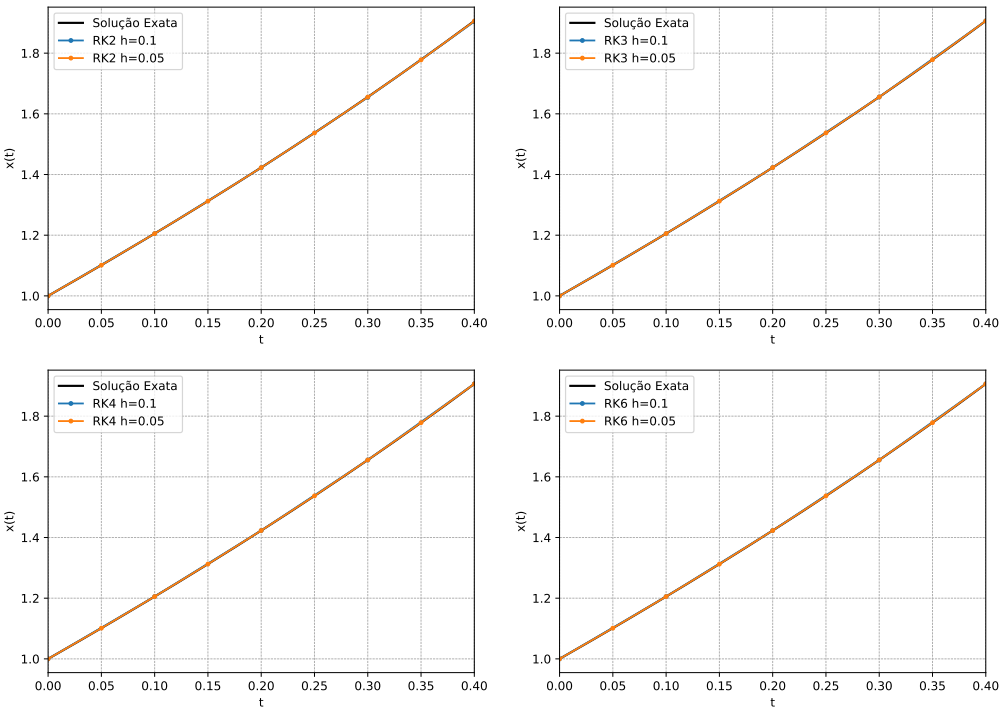

Figura 11: Comparação entre a solução exata e as aproximações obtidas pelo MRK

Método de Multi-Passo (MMP)

Nas tabelas a seguir e no gráfico 12 são apresentados os valores aproximados do PVI (4) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Multi-Passo (MMP) com passo

$h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução analítica. Foi utilizado o método RK6 para obter os passos iniciais necessários para o MMP.

Tabela 71: Resultados da simulação AB para AB2 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350000	0.000000690	0.000057207
0.200000000	1.422956174	1.421955000	0.001001174	0.070358766
0.300000000	1.655529700	1.653006500	0.002523200	0.152410440
0.400000000	1.906385232	1.901712950	0.004672282	0.245085938

Tabela 72: Resultados da simulação AB para AB3 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350687	0.000000002	0.000000177
0.200000000	1.422956174	1.422956169	0.000000005	0.000000366
0.300000000	1.655529700	1.655329184	0.000200516	0.012111895
0.400000000	1.906385232	1.905862950	0.000522282	0.027396455

Tabela 73: Resultados da simulação AB para AB4 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350687	0.000000002	0.000000177
0.200000000	1.422956174	1.422956169	0.000000005	0.000000366
0.300000000	1.655529700	1.655529691	0.000000010	0.000000577
0.400000000	1.906385232	1.906343811	0.000041421	0.002172767

Tabela 74: Resultados da simulação AB para AB6 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350687	0.000000002	0.000000177
0.200000000	1.422956174	1.422956169	0.000000005	0.000000366
0.300000000	1.655529700	1.655529691	0.000000010	0.000000577
0.400000000	1.906385232	1.906385217	0.000000016	0.000000815

Tabela 75: Resultados da simulação AB para AB2 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205236615	0.000114075	0.009464047
0.200000000	1.422956174	1.422526761	0.000429413	0.030177534
0.300000000	1.655529700	1.654650874	0.000878826	0.053084273
0.400000000	1.906385232	1.904879377	0.001505855	0.078990086

Tabela 76: Resultados da simulação AB para AB3 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350690	0.000000000	0.000000003
0.200000000	1.422956174	1.422931305	0.000024869	0.001747697
0.300000000	1.655529700	1.655468595	0.000061106	0.003690996
0.400000000	1.906385232	1.906273125	0.000112107	0.005880587

Tabela 77: Resultados da simulação AB para AB4 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350690	0.000000000	0.000000003
0.200000000	1.422956174	1.422955114	0.000001060	0.000074524
0.300000000	1.655529700	1.655525623	0.000004077	0.000246271
0.400000000	1.906385232	1.906376938	0.000008294	0.000435047

Tabela 78: Resultados da simulação AB para AB6 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.205350690	1.205350690	0.000000000	0.000000003
0.200000000	1.422956174	1.422956174	0.000000000	0.000000006
0.300000000	1.655529700	1.655529689	0.000000011	0.000000651
0.400000000	1.906385232	1.906385192	0.000000040	0.000002122

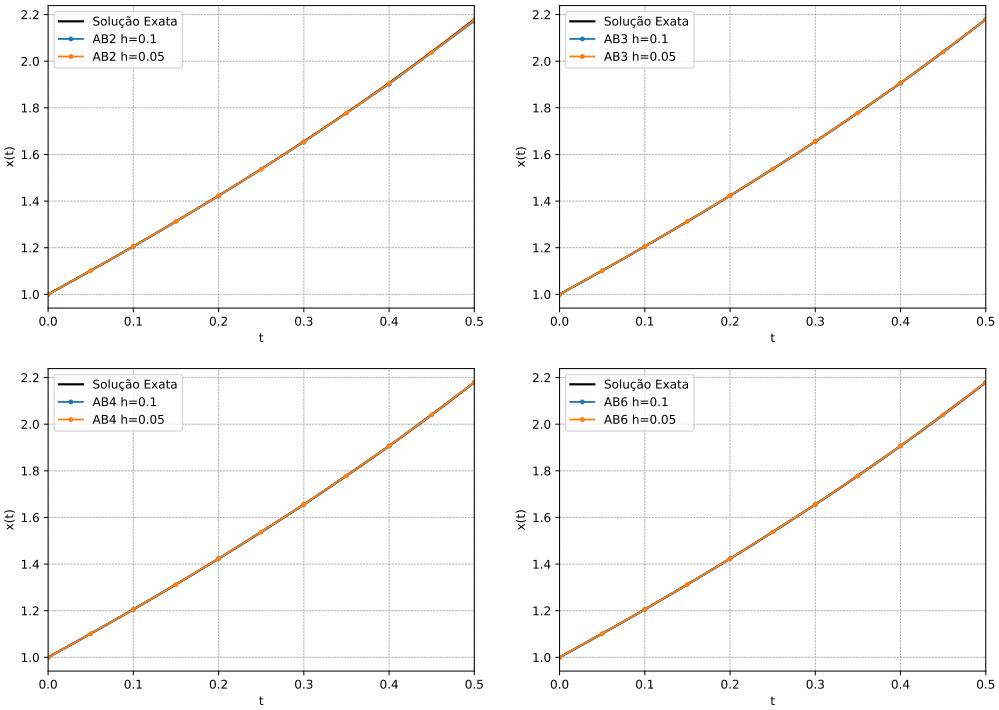

Figura 12: Comparação entre a solução exata e as aproximações obtidas pelo MMP

Exercício 1 - letra D

O Problema de Valor Inicial (PVI) a ser resolvido é dado por:

$$\begin{cases} x'(t) = 2t + \mathrm{e}^{-tx(t)} \\ x(0) = 1 \end{cases} \tag{6}$$

Solução Analítica

A Equação Diferencial Ordinária (EDO) $x'(t) = 2t + e^{-tx(t)}$ é não linear devido ao termo exponencial e não possui uma solução analítica conhecida. Portanto, a análise numérica será realizada comparando os resultados com uma solução de referência, gerada pelo método de Runge-Kutta de 6ª ordem (RK6) com um passo de simulação $h = 0.001$.

Método de Euler Explícito (MEE)

Nas tabelas a seguir e no gráfico 13 são apresentados os valores aproximados do PVI (6) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Euler Explícito (MEE) com passo $h = 0.1$ e $h = 0.05$ comparados com a solução aproximada (RK6 com $h = 0.001$).

Tabela 79: Resultados da simulação MEE para h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.100000000	0.004843021	0.438344728
0.200000000	1.218841063	1.209583414	0.009257650	0.759545289
0.300000000	1.341467896	1.328095572	0.013372323	0.996842598
0.400000000	1.472616839	1.455232989	0.017383850	1.180473421

Tabela 80: Resultados da simulação MEE para h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.102442716	0.002400305	0.217253043
0.200000000	1.218841063	1.214255830	0.004585234	0.376196205
0.300000000	1.341467896	1.334841967	0.006625929	0.493931263
0.400000000	1.472616839	1.463991895	0.008624945	0.585688307

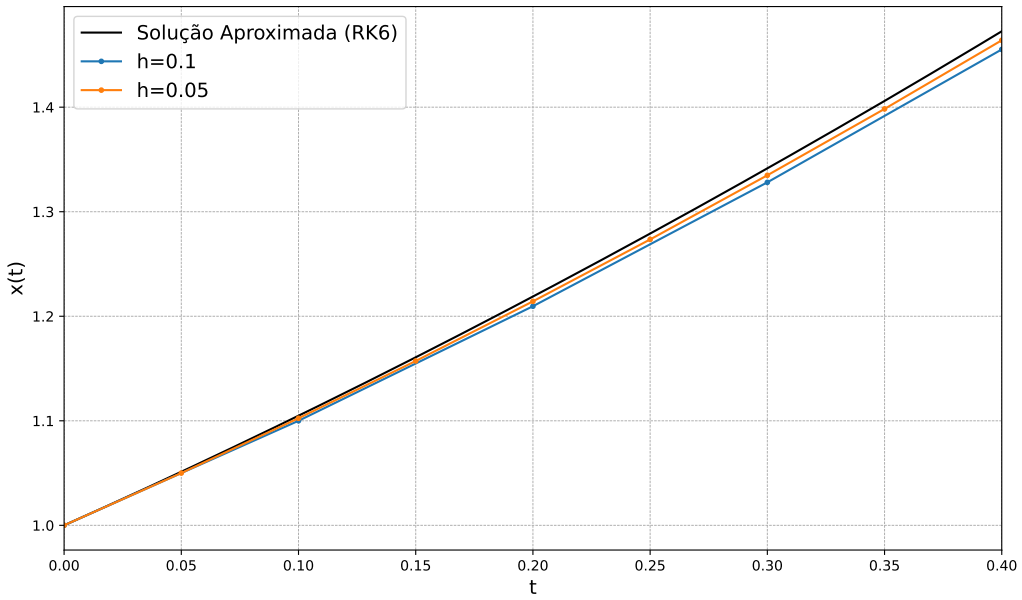

Figura 13: Comparação entre a solução aproximada (RK6 com $h = 0.001$) e as aproximações obtidas pelo MEE

Método da Série de Taylor (MST)

Nas tabelas a seguir e no gráfico 14 são apresentados os valores aproximados do PVI (6) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método da Série de Taylor (MST) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução aproximada (RK6 com $h = 0.001$).

Tabela 81: Resultados da Série de Taylor ordem 2 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.105000000	0.000156979	0.014208250
0.200000000	1.218841063	1.219101227	0.000260163	0.021345126
0.300000000	1.341467896	1.341759938	0.000292042	0.021770312
0.400000000	1.472616839	1.472864653	0.000247813	0.016828071

Tabela 82: Resultados da Série de Taylor ordem 3 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104833333	0.000009688	0.000876849
0.200000000	1.218841063	1.218816215	0.000024848	0.002038672
0.300000000	1.341467896	1.341425217	0.000042679	0.003181532
0.400000000	1.472616839	1.472556452	0.000060387	0.004100691

Tabela 83: Resultados da Série de Taylor ordem 4 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104841667	0.000001354	0.000122594
0.200000000	1.218841063	1.218838836	0.000002227	0.000182733
0.300000000	1.341467896	1.341465366	0.000002530	0.000188597
0.400000000	1.472616839	1.472614525	0.000002315	0.000157185

Tabela 84: Resultados da Série de Taylor ordem 6 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104843026	0.000000005	0.000000475
0.200000000	1.218841063	1.218841071	0.000000007	0.000000596
0.300000000	1.341467896	1.341467902	0.000000006	0.000000456
0.400000000	1.472616839	1.472616843	0.000000003	0.000000212

Tabela 85: Resultados da Série de Taylor ordem 2 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104880782	0.000037761	0.003417740
0.200000000	1.218841063	1.218902514	0.000061450	0.005041711
0.300000000	1.341467896	1.341535021	0.000067125	0.005003880
0.400000000	1.472616839	1.472670803	0.000053963	0.003664461

Tabela 86: Resultados da Série de Taylor ordem 3 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104841695	0.000001327	0.000120066
0.200000000	1.218841063	1.218837779	0.000003284	0.000269467
0.300000000	1.341467896	1.341462377	0.000005519	0.000411400
0.400000000	1.472616839	1.472609149	0.000007690	0.000522217

Tabela 87: Resultados da Série de Taylor ordem 4 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104842941	0.000000080	0.000007249
0.200000000	1.218841063	1.218840935	0.000000129	0.000010569
0.300000000	1.341467896	1.341467754	0.000000142	0.000010607
0.400000000	1.472616839	1.472616715	0.000000125	0.000008474

Tabela 88: Resultados da Série de Taylor ordem 6 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104843021	0.000000000	0.000000007
0.200000000	1.218841063	1.218841064	0.000000000	0.000000008
0.300000000	1.341467896	1.341467896	0.000000000	0.000000005
0.400000000	1.472616839	1.472616840	0.000000000	0.000000002

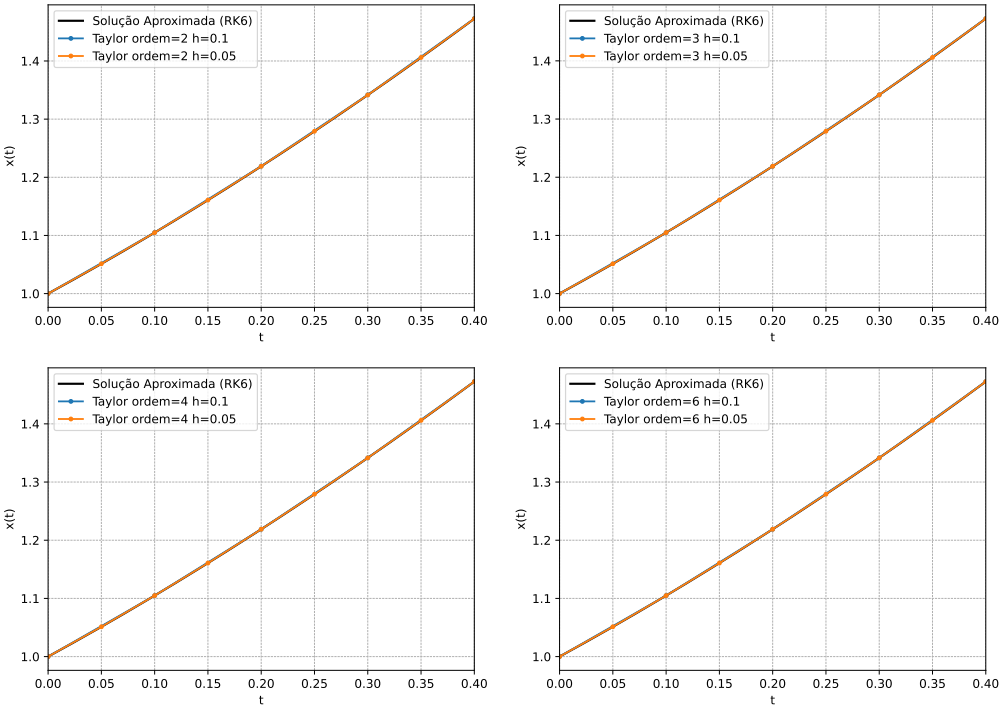

Figura 14: Comparação entre a solução aproximada (RK6 com $h = 0.001$) e as aproximações obtidas pelo MST

Método de Runge-Kutta (MRK)

Nas tabelas a seguir e no gráfico 15 são apresentados os valores aproximados do PVI (6) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Runge-Kutta (MRK) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução aproximada (RK6 com $h = 0.001$).

Tabela 89: Resultados da simulação RK para RK2 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104791707	0.000051314	0.004644495
0.200000000	1.218841063	1.218780769	0.000060294	0.004946850
0.300000000	1.341467896	1.341442367	0.000025529	0.001903079
0.400000000	1.472616839	1.472667549	0.000050710	0.003443511

Tabela 90: Resultados da simulação RK para RK3 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104839609	0.000003412	0.000308835
0.200000000	1.218841063	1.218835360	0.000005704	0.000467975
0.300000000	1.341467896	1.341460842	0.000007053	0.000525796
0.400000000	1.472616839	1.472609222	0.000007618	0.000517304

Tabela 91: Resultados da simulação RK para RK4 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104843052	0.000000031	0.000002833
0.200000000	1.218841063	1.218841129	0.000000065	0.000005362
0.300000000	1.341467896	1.341467988	0.000000092	0.000006846
0.400000000	1.472616839	1.472616942	0.000000103	0.000006965

Tabela 92: Resultados da simulação RK para RK6 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104843021	0.000000000	0.000000001
0.200000000	1.218841063	1.218841063	0.000000000	0.000000002
0.300000000	1.341467896	1.341467896	0.000000000	0.000000002
0.400000000	1.472616839	1.472616839	0.000000000	0.000000003

Tabela 93: Resultados da simulação RK para RK2 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104828923	0.000014098	0.001276045
0.200000000	1.218841063	1.218823869	0.000017194	0.001410680
0.300000000	1.341467896	1.341458889	0.000009007	0.000671425
0.400000000	1.472616839	1.472626670	0.000009831	0.000667586

Tabela 94: Resultados da simulação RK para RK3 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104842585	0.000000437	0.000039510
0.200000000	1.218841063	1.218840329	0.000000735	0.000060271
0.300000000	1.341467896	1.341466980	0.000000916	0.000068268
0.400000000	1.472616839	1.472615841	0.000000998	0.000067791

Tabela 95: Resultados da simulação RK para RK4 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104843023	0.000000002	0.000000171
0.200000000	1.218841063	1.218841067	0.000000004	0.000000327
0.300000000	1.341467896	1.341467902	0.000000006	0.000000421
0.400000000	1.472616839	1.472616846	0.000000006	0.000000431

Tabela 96: Resultados da simulação RK para RK6 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104843021	0.000000000	0.000000000
0.200000000	1.218841063	1.218841063	0.000000000	0.000000000
0.300000000	1.341467896	1.341467896	0.000000000	0.000000000
0.400000000	1.472616839	1.472616839	0.000000000	0.000000000

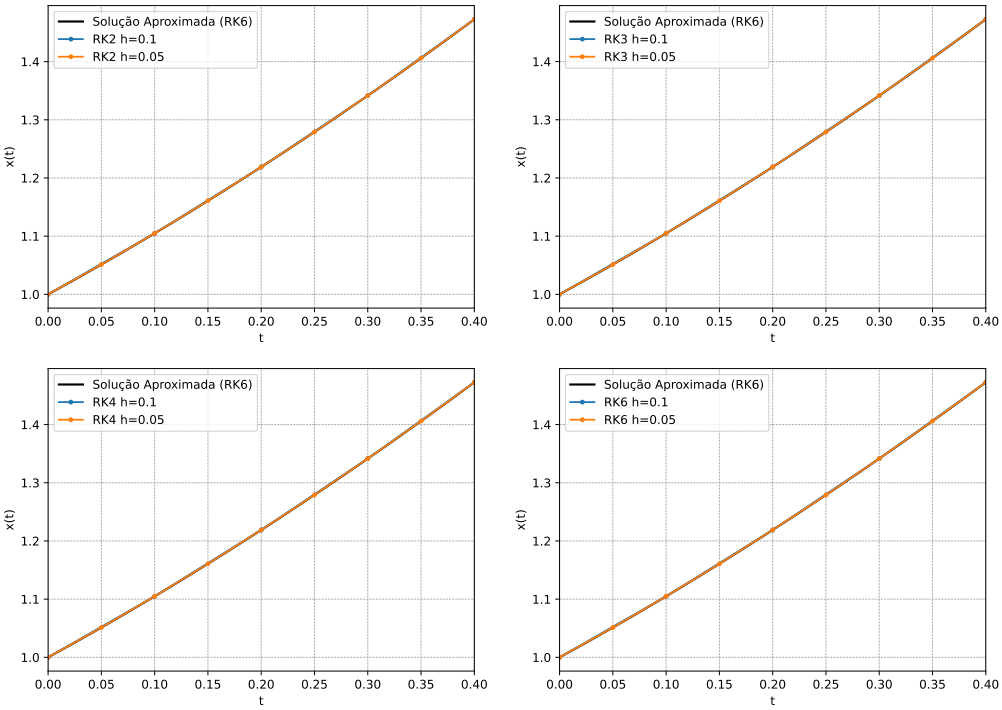

Figura 15: Comparação entre a solução aproximada (RK6 com $h = 0.001$) e as aproximações obtidas pelo MRK

Método de Multi-Passo (MMP)

Nas tabelas a seguir e no gráfico 16 são apresentados os valores aproximados do PVI (6) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Multi-Passo (MMP) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução aproximada (RK6 com $h = 0.001$). Foi utilizado o método RK6 para obter os passos iniciais necessários para o MMP.

Tabela 97: Resultados da simulação AB para AB2 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104843052	0.000000031	0.000002833
0.200000000	1.218841063	1.219153110	0.000312047	0.025601903
0.300000000	1.341467896	1.341926143	0.000458248	0.034160158
0.400000000	1.472616839	1.473034270	0.000417430	0.028346144

Tabela 98: Resultados da simulação AB para AB3 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104843021	0.000000000	0.000000001
0.200000000	1.218841063	1.218841063	0.000000000	0.000000002
0.300000000	1.341467896	1.341324286	0.000143610	0.010705401
0.400000000	1.472616839	1.472313781	0.000303059	0.020579619

Tabela 99: Resultados da simulação AB para AB4 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104843021	0.000000000	0.000000001
0.200000000	1.218841063	1.218841063	0.000000000	0.000000002
0.300000000	1.341467896	1.341467896	0.000000000	0.000000002
0.400000000	1.472616839	1.472597353	0.000019486	0.001323249

Tabela 100: Resultados da simulação AB para AB6 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104843021	0.000000000	0.000000001
0.200000000	1.218841063	1.218841063	0.000000000	0.000000002
0.300000000	1.341467896	1.341467896	0.000000000	0.000000002
0.400000000	1.472616839	1.472616839	0.000000000	0.000000003

Tabela 101: Resultados da simulação AB para AB2 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104889430	0.000046409	0.004200521
0.200000000	1.218841063	1.218953818	0.000112755	0.009250985
0.300000000	1.341467896	1.341603332	0.000135436	0.010096111
0.400000000	1.472616839	1.472728090	0.000111251	0.007554639

Tabela 102: Resultados da simulação AB para AB3 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104843021	0.000000000	0.000000000
0.200000000	1.218841063	1.218825322	0.000015742	0.001291522
0.300000000	1.341467896	1.341432716	0.000035180	0.002622515
0.400000000	1.472616839	1.472561661	0.000055179	0.003746988

Tabela 103: Resultados da simulação AB para AB4 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104843021	0.000000000	0.000000000
0.200000000	1.218841063	1.218839773	0.000001291	0.000105880
0.300000000	1.341467896	1.341465117	0.000002779	0.000207130
0.400000000	1.472616839	1.472613976	0.000002863	0.000194425

Tabela 104: Resultados da simulação AB para AB6 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	1.104843021	1.104843021	0.000000000	0.000000000
0.200000000	1.218841063	1.218841063	0.000000000	0.000000000
0.300000000	1.341467896	1.341467907	0.000000011	0.000000851
0.400000000	1.472616839	1.472616822	0.000000018	0.000001196

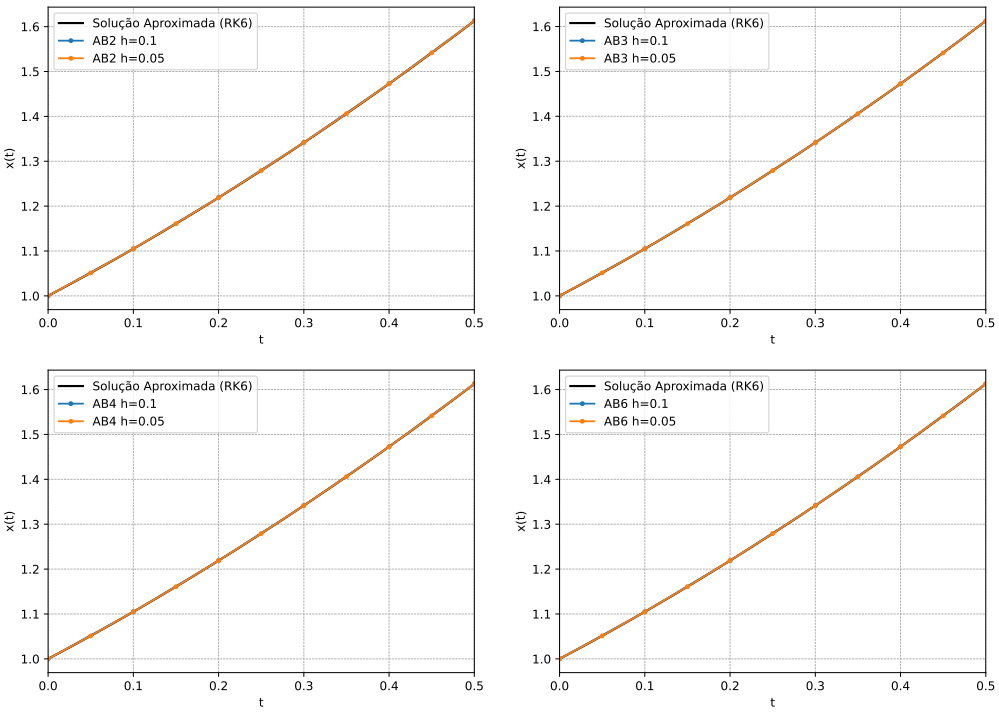

Figura 16: Comparação entre a solução aproximada (RK6 com $h = 0.001$) e as aproximações obtidas pelo MMP

Exercício 1 - letra E

O Problema de Valor Inicial (PVI) a ser resolvido é dado por:

$$\begin{cases} x'(t) = \dfrac{x^2(t) + 2tx(t)}{3 + t^2} \\ x(0) = 0.5 \end{cases} \tag{7}$$

Solução Analítica

A Equação Diferencial Ordinária (EDO) $x'(t) - \left(\frac{2t}{3+t^2}\right) x = \left(\frac{1}{3+t^2}\right) x^2$ é uma Equação de Bernoulli com $n = 2$. A substituição $u = x^{-1}$ a transforma na EDO linear $u' + \left(\frac{2t}{3+t^2}\right) u = -\frac{1}{3+t^2}$.

A solução desta EDO linear para $u(t)$, obtida com o fator integrante $I(t) = 3 + t^2$, é $u(t) = \frac{C-t}{3+t^2}$. Substituindo de volta para $x(t)$ e aplicando a condição inicial $x(0) = 0.5$, determina-se a constante $C = 6$. Portanto, a solução exata para o PVI é:

$$x(t) = \frac{t^2 + 3}{6 - t} \tag{8}$$

Método de Euler Explícito (MEE)

Nas tabelas a seguir e no gráfico 17 são apresentados os valores aproximados do PVI (7) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Euler Explícito (MEE) com passo $h = 0.1$ e $h = 0.05$ comparados com a solução analítica.

Tabela 105: Resultados da simulação MEE para h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.508333333	0.001836158	0.359911406
0.200000000	0.524137931	0.520295773	0.003842158	0.733043243
0.300000000	0.542105263	0.536046629	0.006058635	1.117612191
0.400000000	0.564285714	0.555754524	0.008531190	1.511856513

Tabela 106: Resultados da simulação MEE para h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.509239118	0.000930374	0.182365601
0.200000000	0.524137931	0.522187251	0.001950680	0.372169181
0.300000000	0.542105263	0.539022947	0.003082316	0.568582622
0.400000000	0.564285714	0.559936345	0.004349369	0.770774305

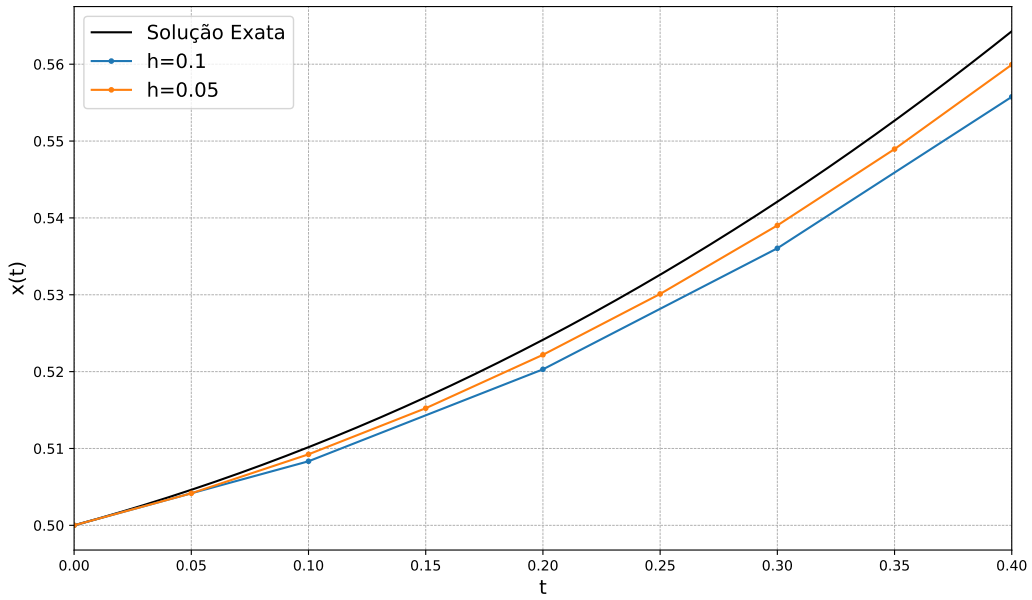

Figura 17: Comparação entre a solução exata e as aproximações obtidas pelo MEE

Método da Série de Taylor (MST)

Nas tabelas a seguir e no gráfico 18 são apresentados os valores aproximados do PVI (7) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método da Série de Taylor (MST) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução analítica.

Tabela 107: Resultados da Série de Taylor ordem 2 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510138889	0.000030603	0.005998523
0.200000000	0.524137931	0.524073213	0.000064718	0.012347590
0.300000000	0.542105263	0.542002094	0.000103169	0.019031180
0.400000000	0.564285714	0.564138817	0.000146897	0.026032361

Tabela 108: Resultados da Série de Taylor ordem 3 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510168981	0.000000510	0.000099975
0.200000000	0.524137931	0.524136843	0.000001088	0.000207571
0.300000000	0.542105263	0.542103514	0.000001750	0.000322743
0.400000000	0.564285714	0.564283201	0.000002513	0.000445422

Tabela 109: Resultados da Série de Taylor ordem 4 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169483	0.000000009	0.000001666
0.200000000	0.524137931	0.524137913	0.000000018	0.000003489
0.300000000	0.542105263	0.542105233	0.000000030	0.000005474
0.400000000	0.564285714	0.564285671	0.000000043	0.000007623

Tabela 110: Resultados da Série de Taylor ordem 6 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169492	0.000000000	0.000000000
0.200000000	0.524137931	0.524137931	0.000000000	0.000000001
0.300000000	0.542105263	0.542105263	0.000000000	0.000000002
0.400000000	0.564285714	0.564285714	0.000000000	0.000000002

Tabela 111: Resultados da Série de Taylor ordem 2 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510161702	0.000007790	0.001526924
0.200000000	0.524137931	0.524121440	0.000016491	0.003146284
0.300000000	0.542105263	0.542078948	0.000026315	0.004854237
0.400000000	0.564285714	0.564248208	0.000037506	0.006646623

Tabela 112: Resultados da Série de Taylor ordem 3 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169426	0.000000065	0.000012778
0.200000000	0.524137931	0.524137792	0.000000139	0.000026558
0.300000000	0.542105263	0.542105039	0.000000224	0.000041335
0.400000000	0.564285714	0.564285392	0.000000322	0.000057103

Tabela 113: Resultados da Série de Taylor ordem 4 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169491	0.000000001	0.000000107
0.200000000	0.524137931	0.524137930	0.000000001	0.000000224
0.300000000	0.542105263	0.542105261	0.000000002	0.000000352
0.400000000	0.564285714	0.564285712	0.000000003	0.000000491

Tabela 114: Resultados da Série de Taylor ordem 6 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169492	0.000000000	0.000000000
0.200000000	0.524137931	0.524137931	0.000000000	0.000000000
0.300000000	0.542105263	0.542105263	0.000000000	0.000000000
0.400000000	0.564285714	0.564285714	0.000000000	0.000000000

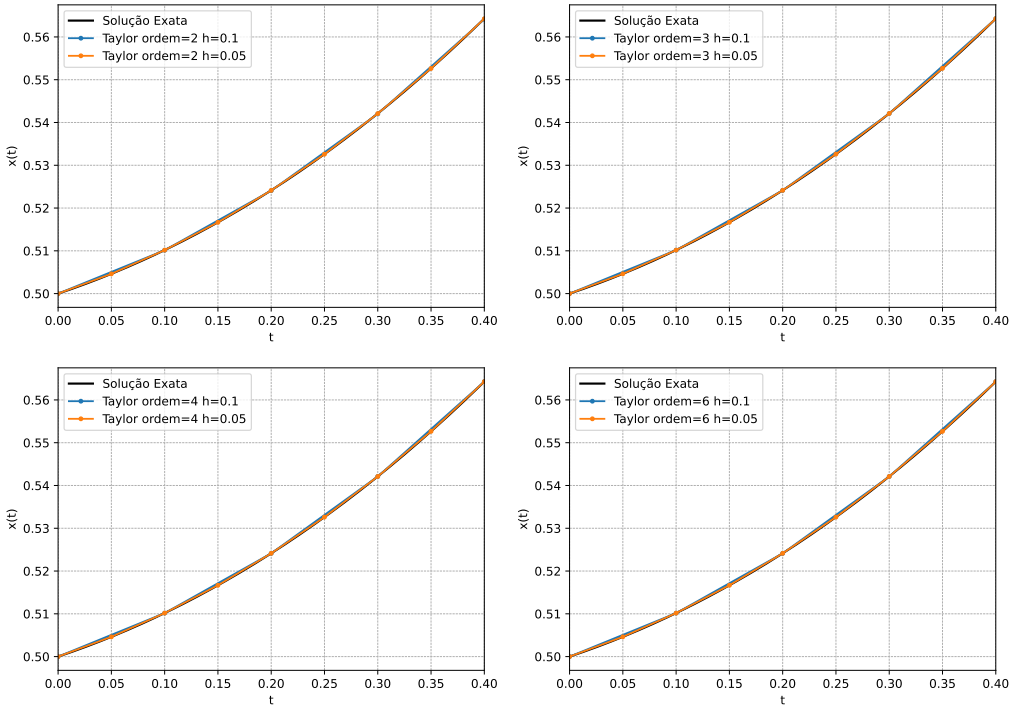

Figura 18: Comparação entre a solução exata e as aproximações obtidas pelo MST

Método de Runge-Kutta (MRK)

Nas tabelas a seguir e no gráfico 19 são apresentados os valores aproximados do PVI (7) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Runge-Kutta (MRK) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução analítica.

Tabela 115: Resultados da simulação RK para RK2 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510147887	0.000021605	0.004234838
0.200000000	0.524137931	0.524086055	0.000051876	0.009897424
0.300000000	0.542105263	0.542013153	0.000092111	0.016991267
0.400000000	0.564285714	0.564141937	0.000143777	0.025479447

Tabela 116: Resultados da simulação RK para RK3 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510170378	0.000000887	0.000173802
0.200000000	0.524137931	0.524139672	0.000001741	0.000332120
0.300000000	0.542105263	0.542107794	0.000002531	0.000466834
0.400000000	0.564285714	0.564288937	0.000003222	0.000571041

Tabela 117: Resultados da simulação RK para RK4 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169500	0.000000008	0.000001664
0.200000000	0.524137931	0.524137950	0.000000019	0.000003549
0.300000000	0.542105263	0.542105293	0.000000030	0.000005513
0.400000000	0.564285714	0.564285756	0.000000042	0.000007388

Tabela 118: Resultados da simulação RK para RK6 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169492	0.000000000	0.000000001
0.200000000	0.524137931	0.524137931	0.000000000	0.000000002
0.300000000	0.542105263	0.542105263	0.000000000	0.000000003
0.400000000	0.564285714	0.564285714	0.000000000	0.000000004

Tabela 119: Resultados da simulação RK para RK2 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510164405	0.000005086	0.000996937
0.200000000	0.524137931	0.524125577	0.000012354	0.002356944
0.300000000	0.542105263	0.542083123	0.000022140	0.004084033
0.400000000	0.564285714	0.564250888	0.000034826	0.006171682

Tabela 120: Resultados da simulação RK para RK3 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169604	0.000000113	0.000022116
0.200000000	0.524137931	0.524138153	0.000000222	0.000042399
0.300000000	0.542105263	0.542105587	0.000000324	0.000059797
0.400000000	0.564285714	0.564286128	0.000000414	0.000073397

Tabela 121: Resultados da simulação RK para RK4 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169492	0.000000001	0.000000107
0.200000000	0.524137931	0.524137932	0.000000001	0.000000229
0.300000000	0.542105263	0.542105265	0.000000002	0.000000357
0.400000000	0.564285714	0.564285717	0.000000003	0.000000481

Tabela 122: Resultados da simulação RK para RK6 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169492	0.000000000	0.000000000
0.200000000	0.524137931	0.524137931	0.000000000	0.000000000
0.300000000	0.542105263	0.542105263	0.000000000	0.000000000
0.400000000	0.564285714	0.564285714	0.000000000	0.000000000

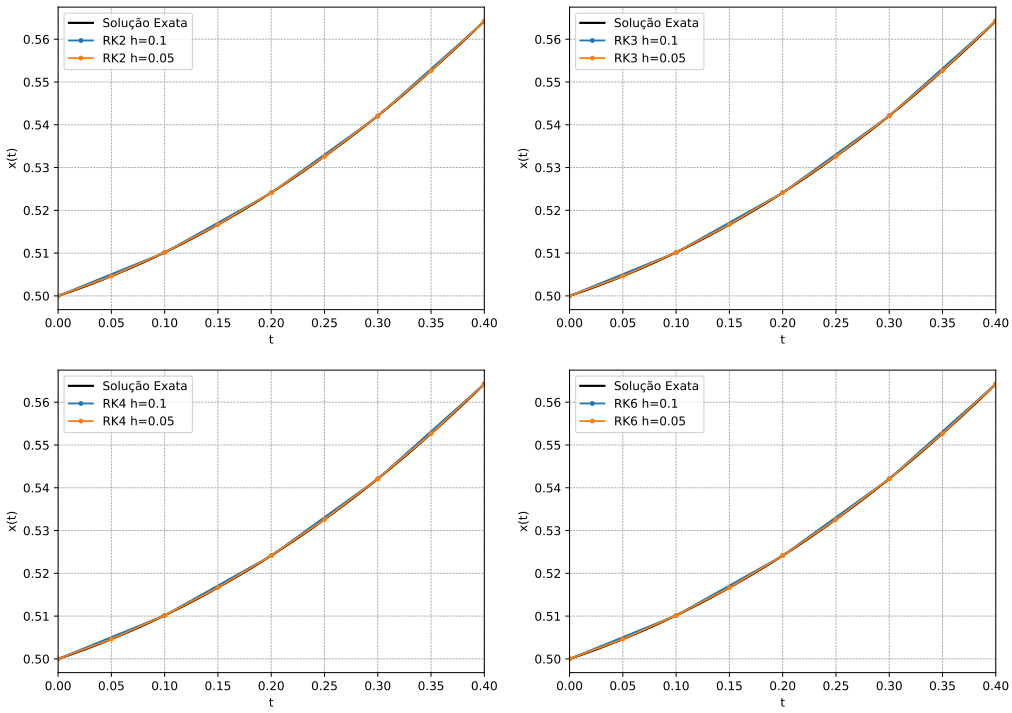

Figura 19: Comparação entre a solução exata e as aproximações obtidas pelo MRK

Método de Multi-Passo (MMP)

Nas tabelas a seguir e no gráfico 20 são apresentados os valores aproximados do PVI (7) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Multi-Passo (MMP) com passo

$h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução analítica. Foi utilizado o método RK6 para obter os passos iniciais necessários para o MMP.

Tabela 123: Resultados da simulação AB para AB2 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169500	0.000000008	0.000001664
0.200000000	0.524137931	0.524057990	0.000079941	0.015251826
0.300000000	0.542105263	0.541934011	0.000171252	0.031590145
0.400000000	0.564285714	0.564010611	0.000275103	0.048752474

Tabela 124: Resultados da simulação AB para AB3 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169492	0.000000000	0.000000001
0.200000000	0.524137931	0.524137931	0.000000000	0.000000002
0.300000000	0.542105263	0.542100166	0.000005097	0.000940250
0.400000000	0.564285714	0.564274529	0.000011185	0.001982202

Tabela 125: Resultados da simulação AB para AB4 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169492	0.000000000	0.000000001
0.200000000	0.524137931	0.524137931	0.000000000	0.000000002
0.300000000	0.542105263	0.542105263	0.000000000	0.000000003
0.400000000	0.564285714	0.564285287	0.000000427	0.000075672

Tabela 126: Resultados da simulação AB para AB6 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169492	0.000000000	0.000000001
0.200000000	0.524137931	0.524137931	0.000000000	0.000000002
0.300000000	0.542105263	0.542105263	0.000000000	0.000000003
0.400000000	0.564285714	0.564285714	0.000000000	0.000000004

Tabela 127: Resultados da simulação AB para AB2 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510159799	0.000009692	0.001899770
0.200000000	0.524137931	0.524106984	0.000030947	0.005904371
0.300000000	0.542105263	0.542050323	0.000054940	0.010134623
0.400000000	0.564285714	0.564203441	0.000082274	0.014580135

Tabela 128: Resultados da simulação AB para AB3 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169492	0.000000000	0.000000000
0.200000000	0.524137931	0.524137306	0.000000625	0.000119214
0.300000000	0.542105263	0.542103921	0.000001342	0.000247517
0.400000000	0.564285714	0.564283544	0.000002170	0.000384625

Tabela 129: Resultados da simulação AB para AB4 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169492	0.000000000	0.000000000
0.200000000	0.524137931	0.524137919	0.000000012	0.000002301
0.300000000	0.542105263	0.542105224	0.000000040	0.000007296
0.400000000	0.564285714	0.564285643	0.000000072	0.000012699

Tabela 130: Resultados da simulação AB para AB6 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.510169492	0.510169492	0.000000000	0.000000000
0.200000000	0.524137931	0.524137931	0.000000000	0.000000000
0.300000000	0.542105263	0.542105263	0.000000000	0.000000007
0.400000000	0.564285714	0.564285714	0.000000000	0.000000021

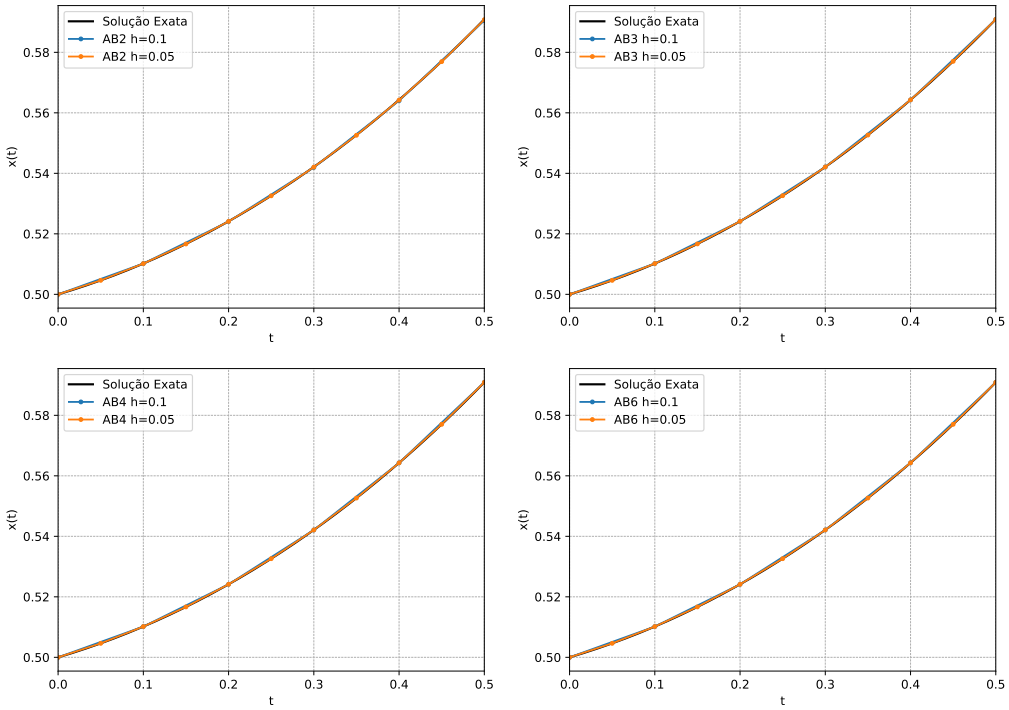

Figura 20: Comparação entre a solução exata e as aproximações obtidas pelo MMP

Exercício 1 - letra F

O Problema de Valor Inicial (PVI) a ser resolvido é dado por:

$$\begin{cases} x'(t) = (t^2 - x^2(t))\text{sen}(x(t)) \\ x(0) = -1 \end{cases} \tag{9}$$

Solução Analítica

A Equação Diferencial Ordinária (EDO) $x'(t) = (t^2 - x^2(t))\sin(x(t))$ é altamente **não linear** devido à combinação dos termos polinomiais e trigonométricos, e não possui uma solução analítica conhecida. Portanto, a análise numérica será realizada comparando os resultados com uma solução de referência, gerada pelo método de Runge-Kutta de 6ª ordem (RK6) com um passo de simulação $h = 0.001$.

Método de Euler Explícito (MEE)

Nas tabelas a seguir e no gráfico 21 são apresentados os valores aproximados do PVI (9) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Euler Explícito (MEE) com passo $h = 0.1$ e $h = 0.05$ comparados com a solução aproximada (RK6 com $h = 0.001$).

Tabela 131: Resultados da simulação MEE para h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488014362	0.000558071	0.114224791
0.200000000	0.478785654	0.477316649	0.001469005	0.306818933
0.300000000	0.471368732	0.468687733	0.002680999	0.568768849
0.400000000	0.467003641	0.462830415	0.004173227	0.893617624

Tabela 132: Resultados da simulação MEE para h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488280704	0.000291728	0.059710354
0.200000000	0.478785654	0.478030587	0.000755067	0.157704636
0.300000000	0.471368732	0.470002041	0.001366690	0.289940834
0.400000000	0.467003641	0.464885134	0.002118508	0.453638333

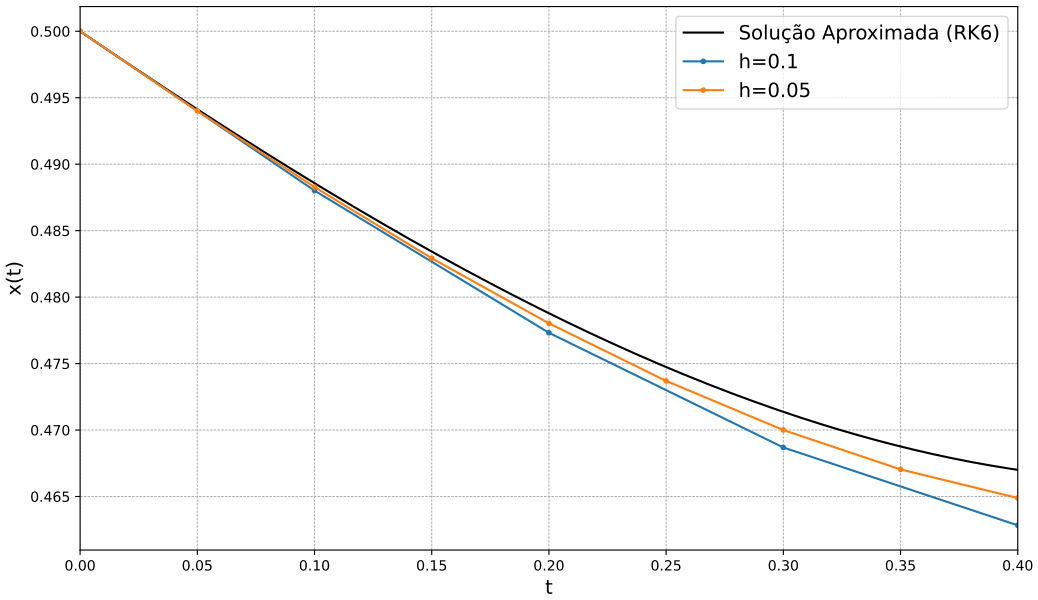

Figura 21: Comparação entre a solução aproximada (RK6 com $h = 0.001$) e as aproximações obtidas pelo MEE

Método da Série de Taylor (MST)

Nas tabelas a seguir e no gráfico 22 são apresentados os valores aproximados do PVI (9) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método da Série de Taylor (MST) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução aproximada (RK6 com $h = 0.001$).

Tabela 133: Resultados da Série de Taylor ordem 2 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488433152	0.000139280	0.028507531
0.200000000	0.478785654	0.478531114	0.000254540	0.053163703
0.300000000	0.471368732	0.471014203	0.000354529	0.075212581
0.400000000	0.467003641	0.466555193	0.000448449	0.096026828

Tabela 134: Resultados da Série de Taylor ordem 3 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488576995	0.000004562	0.000933772
0.200000000	0.478785654	0.478793178	0.000007524	0.001571403
0.300000000	0.471368732	0.471377494	0.000008762	0.001858848
0.400000000	0.467003641	0.467011839	0.000008197	0.001755295

Tabela 135: Resultados da Série de Taylor ordem 4 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572203	0.000000229	0.000046876
0.200000000	0.478785654	0.478785148	0.000000506	0.000105681
0.300000000	0.471368732	0.471367917	0.000000814	0.000172753
0.400000000	0.467003641	0.467002491	0.000001150	0.000246297

Tabela 136: Resultados da Série de Taylor ordem 6 com h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572433	0.000000001	0.000000151
0.200000000	0.478785654	0.478785655	0.000000001	0.000000227
0.300000000	0.471368732	0.471368733	0.000000001	0.000000248
0.400000000	0.467003641	0.467003643	0.000000001	0.000000236

Tabela 137: Resultados da Série de Taylor ordem 2 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488538729	0.000033703	0.006898317
0.200000000	0.478785654	0.478723906	0.000061748	0.012896750
0.300000000	0.471368732	0.471282436	0.000086296	0.018307474
0.400000000	0.467003641	0.466894017	0.000109624	0.023473953

Tabela 138: Resultados da Série de Taylor ordem 3 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572970	0.000000537	0.000110011
0.200000000	0.478785654	0.478786529	0.000000875	0.000182697
0.300000000	0.471368732	0.471369729	0.000000998	0.000211621
0.400000000	0.467003641	0.467004537	0.000000895	0.000191751

Tabela 139: Resultados da Série de Taylor ordem 4 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572418	0.000000015	0.000003028
0.200000000	0.478785654	0.478785622	0.000000032	0.000006747
0.300000000	0.471368732	0.471368680	0.000000052	0.000010961
0.400000000	0.467003641	0.467003569	0.000000073	0.000015578

Tabela 140: Resultados da Série de Taylor ordem 6 com h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572432	0.000000000	0.000000002
0.200000000	0.478785654	0.478785654	0.000000000	0.000000003
0.300000000	0.471368732	0.471368732	0.000000000	0.000000003
0.400000000	0.467003641	0.467003641	0.000000000	0.000000003

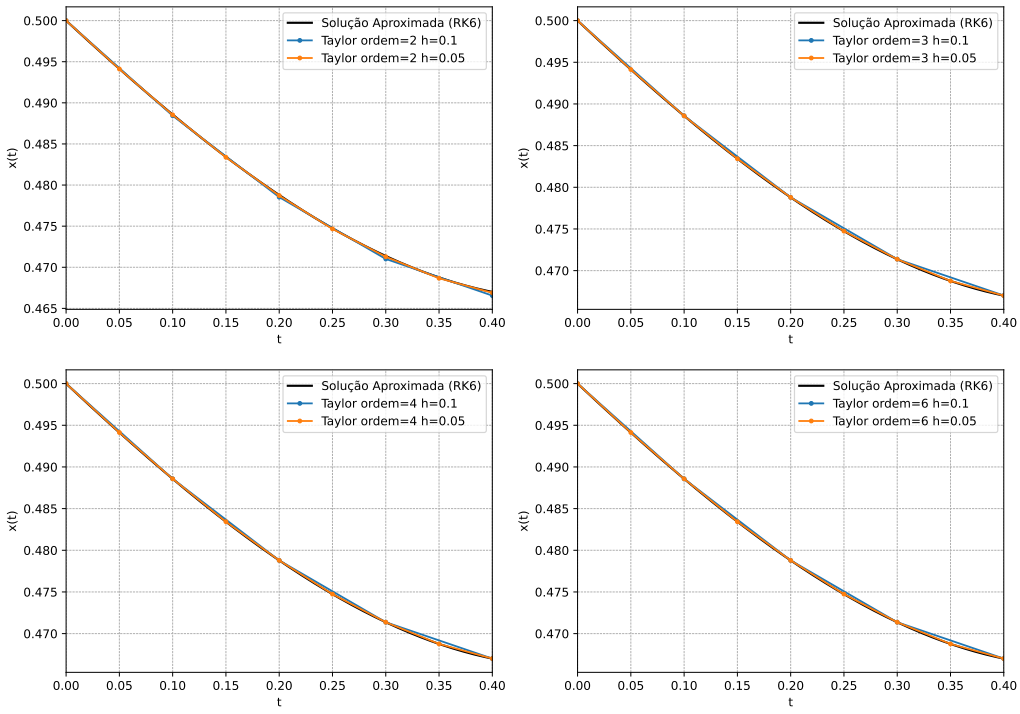

Figura 22: Comparação entre a solução aproximada (RK6 com $h = 0.001$) e as aproximações obtidas pelo MST

Método de Runge-Kutta (MRK)

Nas tabelas a seguir e no gráfico 23 são apresentados os valores aproximados do PVI (9) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Runge-Kutta (MRK) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução aproximada (RK6 com $h = 0.001$).

Tabela 141: Resultados da simulação RK para RK2 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488658324	0.000085892	0.017580212
0.200000000	0.478785654	0.478955531	0.000169877	0.035480859
0.300000000	0.471368732	0.471620993	0.000252261	0.053516750
0.400000000	0.467003641	0.467337949	0.000334308	0.071585644

Tabela 142: Resultados da simulação RK para RK3 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572411	0.000000021	0.000004397
0.200000000	0.478785654	0.478785734	0.000000080	0.000016788
0.300000000	0.471368732	0.471369133	0.000000402	0.000085208
0.400000000	0.467003641	0.467004654	0.000001013	0.000216818

Tabela 143: Resultados da simulação RK para RK4 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572431	0.000000002	0.000000355
0.200000000	0.478785654	0.478785647	0.000000007	0.000001409
0.300000000	0.471368732	0.471368715	0.000000017	0.000003546
0.400000000	0.467003641	0.467003608	0.000000033	0.000007090

Tabela 144: Resultados da simulação RK para RK6 h=0.1 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572432	0.000000000	0.000000001
0.200000000	0.478785654	0.478785654	0.000000000	0.000000002
0.300000000	0.471368732	0.471368732	0.000000000	0.000000002
0.400000000	0.467003641	0.467003641	0.000000000	0.000000001

Tabela 145: Resultados da simulação RK para RK2 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488593903	0.000021471	0.004394652
0.200000000	0.478785654	0.478828174	0.000042520	0.008880777
0.300000000	0.471368732	0.471431994	0.000063263	0.013421053
0.400000000	0.467003641	0.467087687	0.000084045	0.017996715

Tabela 146: Resultados da simulação RK para RK3 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572429	0.000000003	0.000000686
0.200000000	0.478785654	0.478785663	0.000000009	0.000001930
0.300000000	0.471368732	0.471368782	0.000000050	0.000010604
0.400000000	0.467003641	0.467003770	0.000000128	0.000027446

Tabela 147: Resultados da simulação RK para RK4 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572432	0.000000000	0.000000015
0.200000000	0.478785654	0.478785654	0.000000000	0.000000076
0.300000000	0.471368732	0.471368731	0.000000001	0.000000205
0.400000000	0.467003641	0.467003639	0.000000002	0.000000424

Tabela 148: Resultados da simulação RK para RK6 h=0.05 no intervalo [0, 0.4].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572432	0.000000000	0.000000000
0.200000000	0.478785654	0.478785654	0.000000000	0.000000000
0.300000000	0.471368732	0.471368732	0.000000000	0.000000000
0.400000000	0.467003641	0.467003641	0.000000000	0.000000000

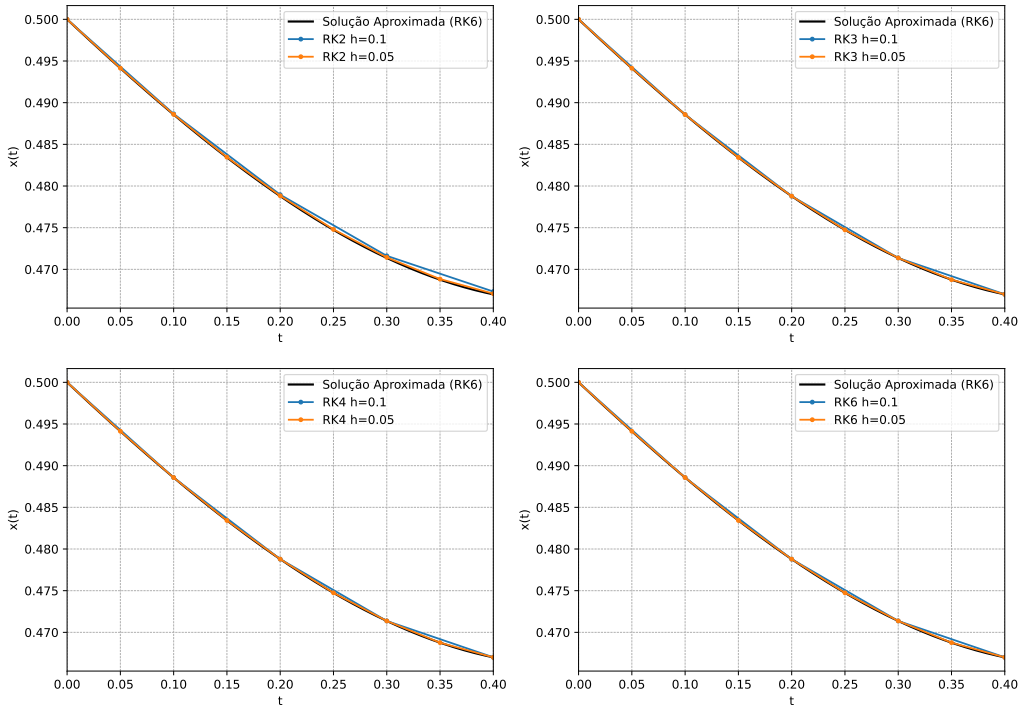

Figura 23: Comparação entre a solução aproximada (RK6 com $h = 0.001$) e as aproximações obtidas pelo MRK

Método de Multi-Passo (MMP)

Nas tabelas a seguir e no gráfico 24 são apresentados os valores aproximados do PVI (9) dado em $t = [0.1, 0.2, 0.3, 0.4]$ utilizando o Método de Multi-Passo (MMP) com passo $h = 0.1$ e $h = 0.05$, e ordens 2, 3, 4 e 6, comparados com a solução aproximada (RK6 com $h = 0.001$). Foi utilizado o método RK6 para obter os passos iniciais necessários para o MMP.

Tabela 149: Resultados da simulação AB para AB2 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572431	0.000000002	0.000000355
0.200000000	0.478785654	0.478463443	0.000322211	0.067297556
0.300000000	0.471368732	0.470782950	0.000585782	0.124272591
0.400000000	0.467003641	0.466175962	0.000827679	0.177231816

Tabela 150: Resultados da simulação AB para AB3 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572432	0.000000000	0.000000001
0.200000000	0.478785654	0.478785654	0.000000000	0.000000002
0.300000000	0.471368732	0.471394623	0.000025891	0.005492766
0.400000000	0.467003641	0.467038060	0.000034419	0.007370129

Tabela 151: Resultados da simulação AB para AB4 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572432	0.000000000	0.000000001
0.200000000	0.478785654	0.478785654	0.000000000	0.000000002
0.300000000	0.471368732	0.471368732	0.000000000	0.000000002
0.400000000	0.467003641	0.466989988	0.000013653	0.002923547

Tabela 152: Resultados da simulação AB para AB6 h=0.1 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572432	0.000000000	0.000000001
0.200000000	0.478785654	0.478785654	0.000000000	0.000000002
0.300000000	0.471368732	0.471368732	0.000000000	0.000000002
0.400000000	0.467003641	0.467003641	0.000000000	0.000000001

Tabela 153: Resultados da simulação AB para AB2 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488530007	0.000042425	0.008683492
0.200000000	0.478785654	0.478670556	0.000115098	0.024039531
0.300000000	0.471368732	0.471190515	0.000178217	0.037808309
0.400000000	0.467003641	0.466766298	0.000237344	0.050822693

Tabela 154: Resultados da simulação AB para AB3 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572432	0.000000000	0.000000000
0.200000000	0.478785654	0.478789532	0.000003878	0.000810059
0.300000000	0.471368732	0.471374620	0.000005888	0.001249185
0.400000000	0.467003641	0.467009560	0.000005919	0.001267352

Tabela 155: Resultados da simulação AB para AB4 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572432	0.000000000	0.000000000
0.200000000	0.478785654	0.478785288	0.000000366	0.000076506
0.300000000	0.471368732	0.471367583	0.000001148	0.000243608
0.400000000	0.467003641	0.467001634	0.000002007	0.000429751

Tabela 156: Resultados da simulação AB para AB6 h=0.05 no intervalo [0, 0.5].

t	x_exato(t)	x(t)	err_loc	err_rel (%)
0.100000000	0.488572432	0.488572432	0.000000000	0.000000000
0.200000000	0.478785654	0.478785654	0.000000000	0.000000000
0.300000000	0.471368732	0.471368735	0.000000004	0.000000788
0.400000000	0.467003641	0.467003648	0.000000007	0.000001455

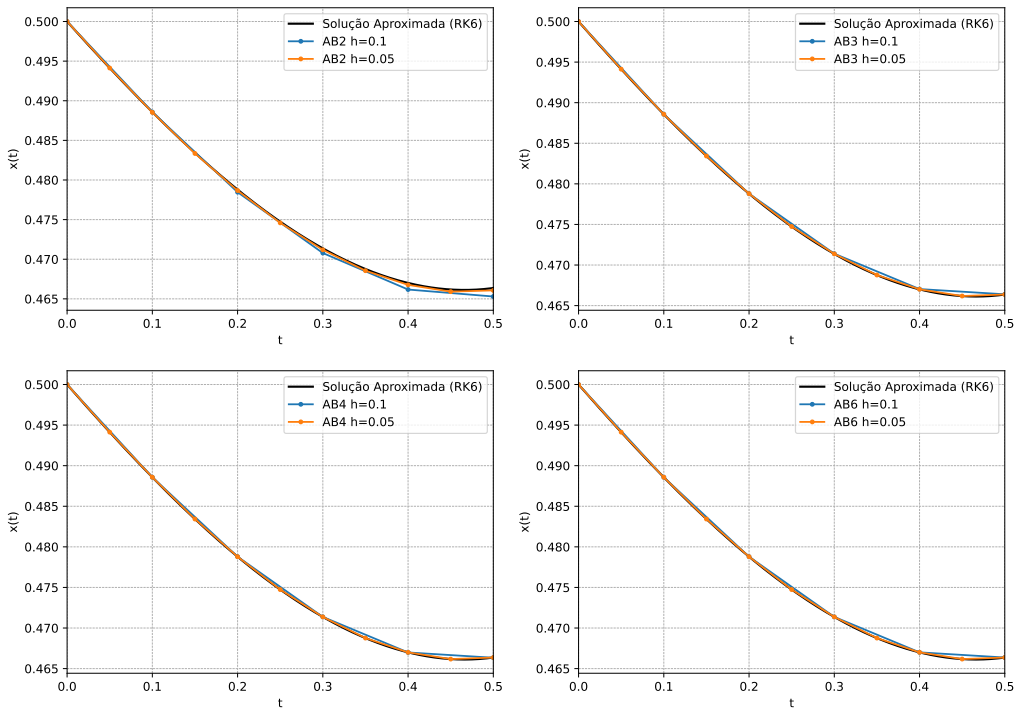

Figura 24: Comparação entre a solução aproximada (RK6 com $h = 0.001$) e as aproximações obtidas pelo MMP

Conclusões

A análise numérica dos Problemas de Valor Inicial (PVIs) propostos permitiu uma comparação detalhada do desempenho de quatro famílias de métodos: Euler Explícito

(MEE), Série de Taylor (MST), Runge-Kutta (MRK) e Adams-Bashforth (MMP).

O MEE, embora seja o mais simples de implementar e de menor custo computacional, demonstrou a menor precisão. O MST, em contrapartida, alcançou altíssima acurácia, como esperado, contudo seu custo computacional reside na complexidade da implementação, que, embora viabilizada pela derivação simbólica com a biblioteca `SymPy`, representa uma desvantagem prática.

Nesse contexto, os métodos de Runge-Kutta destacaram-se como uma alternativa robusta ao MST, oferecendo acurácia comparável sem a necessidade de calcular derivadas, o que torna sua aplicação mais geral. Em contrapartida, os métodos de múltiplos passos de Adams-Bashforth, embora eficientes, demonstraram uma precisão inferior à dos métodos RK de mesma ordem. Essa desvantagem pode ser atribuída à sua dependência de um método de passo simples (MRK de ordem 6) para a inicialização, cujos erros, mesmo que pequenos, são acumulados e propagados nos cálculos subsequentes.

Exercício 2

Nesta questão, foi realizado um estudo numérico para analisar o comportamento de um modelo de crescimento populacional não linear, descrito pelo seguinte Problema de Valor Inicial (PVI):

$$\begin{cases} x'(t) = rx(t)(1 - x(t)) - \dfrac{x^2(t)}{1 + x^2(t)} = f(t, x) \\ x(0) = 0.1 \\ t \in [0, 10] \end{cases} \tag{10}$$

O modelo representa uma população com crescimento logístico (termo $rx(1 - x)$) sujeita a um termo de perda não linear $\left(\text{termo } \frac{x^2}{1+x^2}\right)$, que se intensifica com o aumento da população. Devido à sua complexidade, esta EDO não possui uma solução analítica conhecida, tornando a análise numérica fundamental.

O estudo consistirá na comparação do desempenho de diferentes métodos numéricos. Foram utilizados métodos de passo fixo, incluindo o Euler Explícito (MEE), o Runge-Kutta de 2ª Ordem (RK2) e o Runge-Kutta de 4ª Ordem (RK4), com os seguintes passos de simulação: $h \in \{1, 0.75, 0.5, 0.25, 0.125\}$. Também foi analisado o desempenho de métodos de passo adaptativo, especificamente as implementações Runge-Kutta-Fehlberg (RKF45) e Dormand-Prince (DOPRI54), partindo de um passo inicial $h = 1$. A acurácia e a eficiência de cada método serão avaliadas comparando as soluções obtidas por cada método.

A análise numérica para o caso com taxa de crescimento $r = 2$ é apresentada a seguir. As Figuras 25, 26, 27 e 28 ilustram a convergência das soluções para os métodos MEE, RK2, RK4 e adaptativos, respectivamente, à medida que o passo de simulação h é refinado.

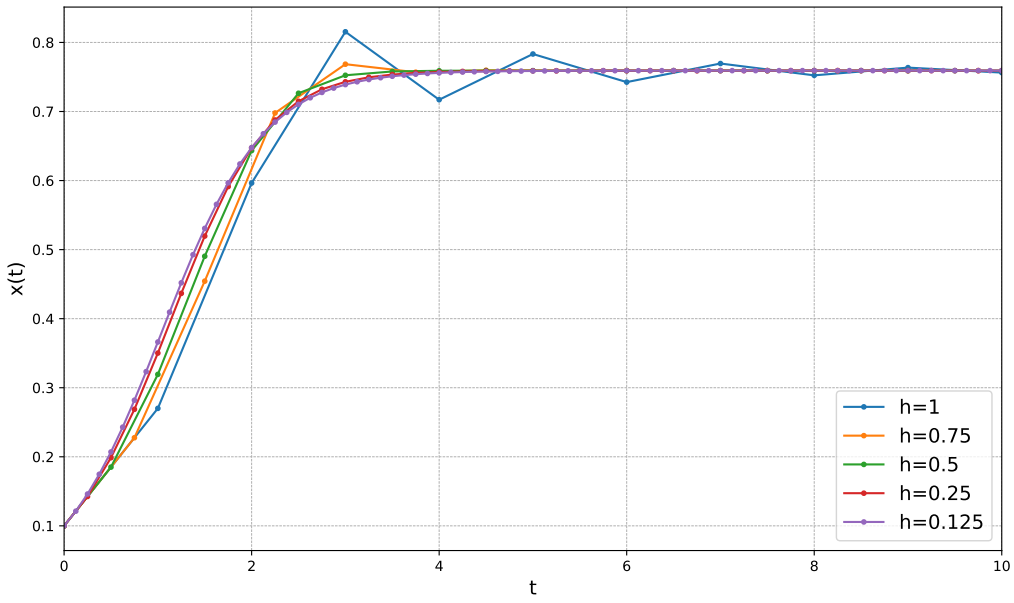

Figura 25: Comparação entre as soluções obtidas pelo MEE

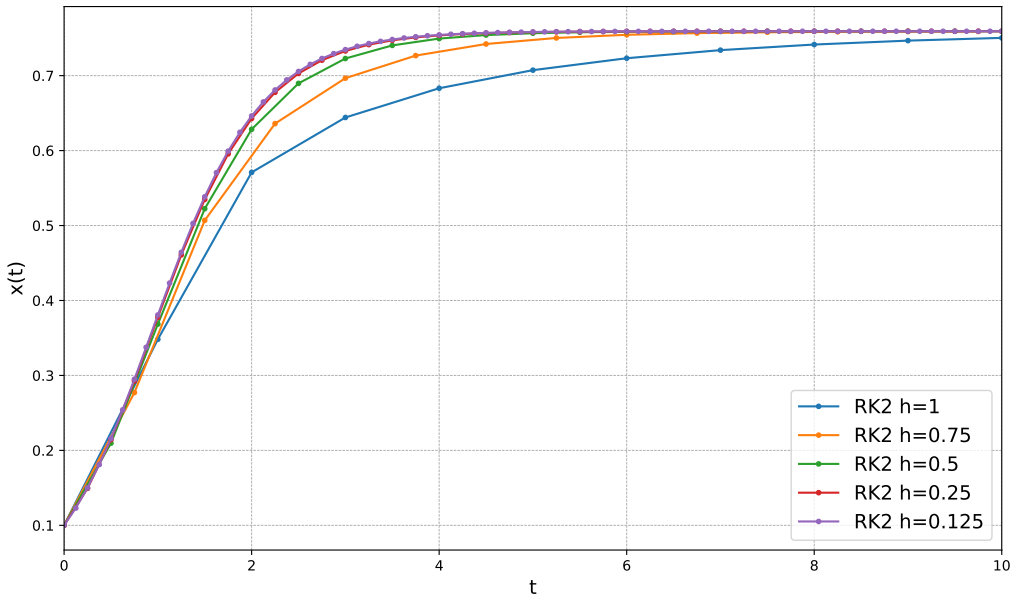

Figura 26: Comparação entre as soluções obtidas pelo MRK de ordem 2

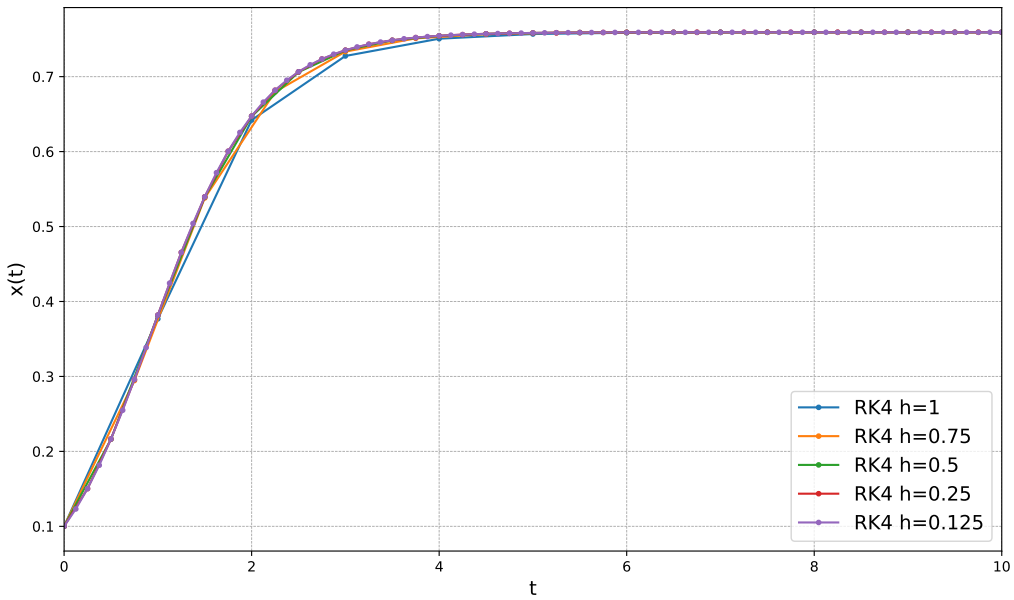

Figura 27: Comparação entre as soluções obtidas pelo MRK de ordem 4

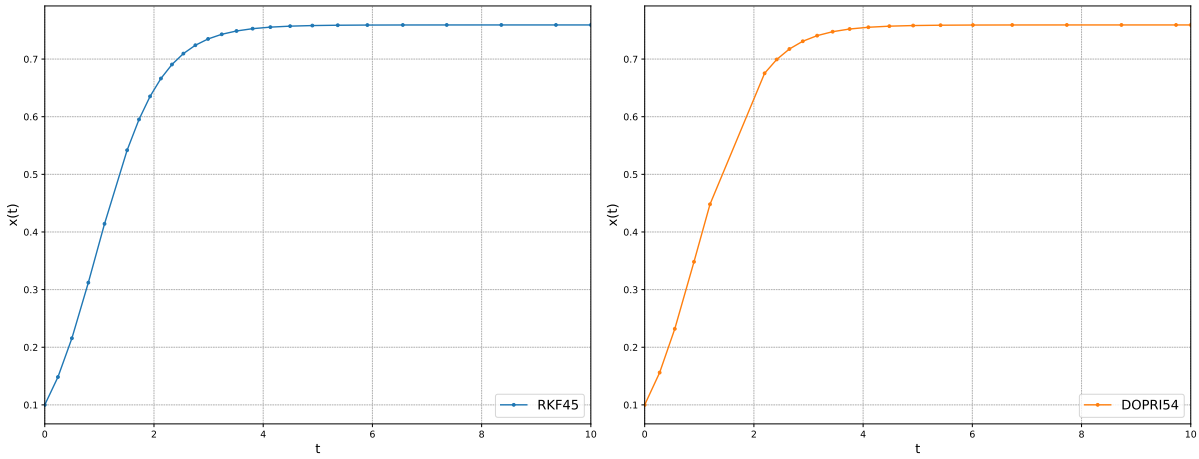

Figura 28: Comparação entre as soluções obtidas pelo MRK adaptativos

Nas tabelas a seguir são apresentados os resultados obtidos por cada método em pontos fixados.

Tabela 157: Comparação dos métodos para h = 1 e r = 2 nos tempos desejados.

t	Euler	RK2	RK4	RKF45	DOPRI54
1.500000000	-	-	-	0.539663753	0.539665446
3.000000000	0.815444335	0.644165186	0.727587309	0.735264861	0.735266146
6.000000000	0.742560385	0.723213278	0.758564071	0.759021243	0.759020702
9.000000000	0.763546544	0.746705880	0.759183530	0.759195013	0.759194750

Tabela 158: Comparação dos métodos para h = 0.75 e r = 2 nos tempos desejados.

t	Euler	RK2	RK4	RKF45	DOPRI54
1.500000000	0.454320650	0.506901532	0.538133401	0.539663753	0.539665446
3.000000000	0.768504677	0.696615459	0.733390958	0.735264861	0.735266146
6.000000000	0.759226173	0.754493307	0.758951701	0.759021243	0.759020702
9.000000000	0.759196247	0.758831945	0.759193884	0.759195013	0.759194750

Tabela 159: Comparação dos métodos para h = 0.5 e r = 2 nos tempos desejados.

t	Euler	RK2	RK4	RKF45	DOPRI54
1.500000000	0.490388592	0.522415210	0.539304126	0.539663753	0.539665446
3.000000000	0.752450829	0.722847714	0.734973122	0.735264861	0.735266146
6.000000000	0.759195945	0.758504369	0.759012114	0.759021243	0.759020702
9.000000000	0.759196154	0.759183168	0.759194791	0.759195013	0.759194750

Tabela 160: Comparação dos métodos para h = 0.25 e r = 2 nos tempos desejados.

t	Euler	RK2	RK4	RKF45	DOPRI54
1.500000000	0.519608972	0.534790881	0.539638456	0.539663753	0.539665446
3.000000000	0.742979646	0.732910536	0.735249918	0.735264861	0.735266146
6.000000000	0.759167567	0.758963770	0.759020537	0.759021243	0.759020702
9.000000000	0.759196106	0.759194151	0.759194900	0.759195013	0.759194750

Tabela 161: Comparação dos métodos para h = 0.125 e r = 2 nos tempos desejados.

t	Euler	RK2	RK4	RKF45	DOPRI54
1.500000000	0.530731799	0.538374241	0.539662885	0.539663753	0.539665446
3.000000000	0.738964247	0.734733377	0.735263991	0.735264861	0.735266146
6.000000000	0.759114060	0.759009606	0.759020914	0.759021243	0.759020702
9.000000000	0.759195831	0.759194769	0.759194905	0.759195013	0.759194750

A análise numérica para o caso com taxa de crescimento $r = 3$ é apresentada a seguir. As Figuras 29, 30, 31 e 32 ilustram a convergência das soluções para os métodos MEE, RK2, RK4 e adaptativos, respectivamente, à medida que o passo de simulação h é refinado.

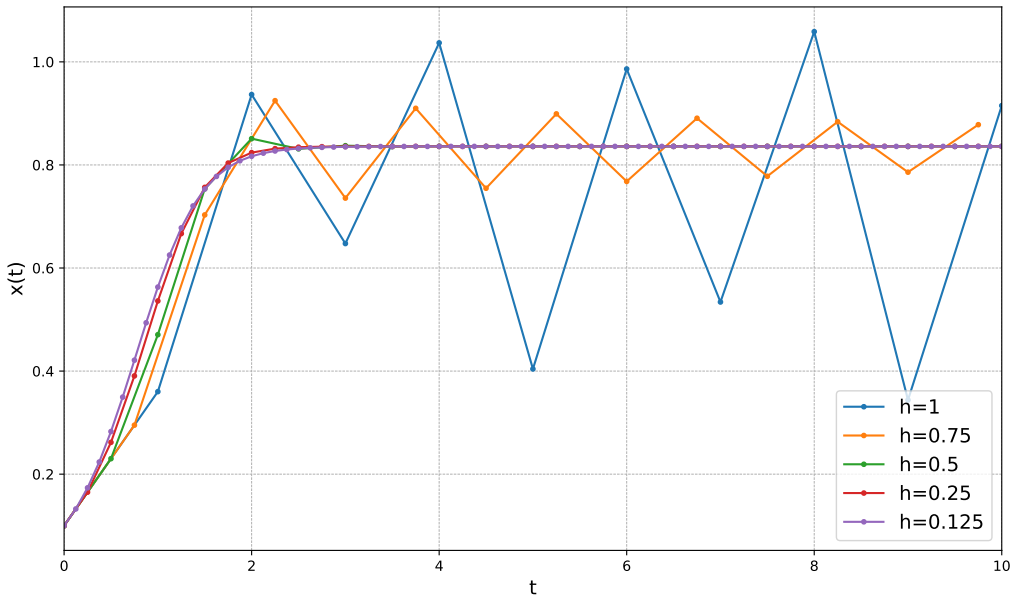

Figura 29: Comparação entre as soluções obtidas pelo MEE

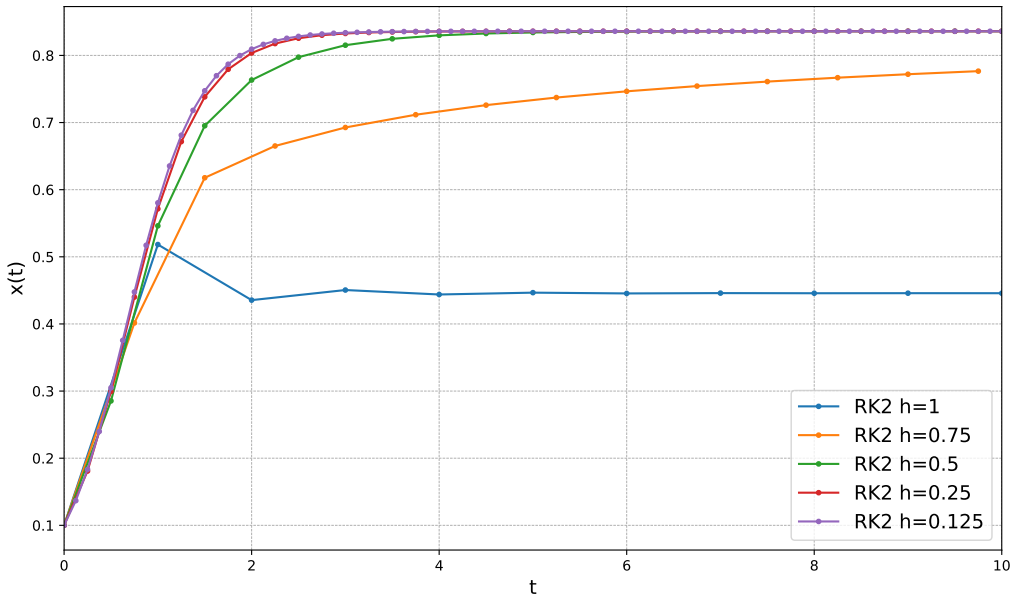

Figura 30: Comparação entre as soluções obtidas pelo MRK de ordem 2

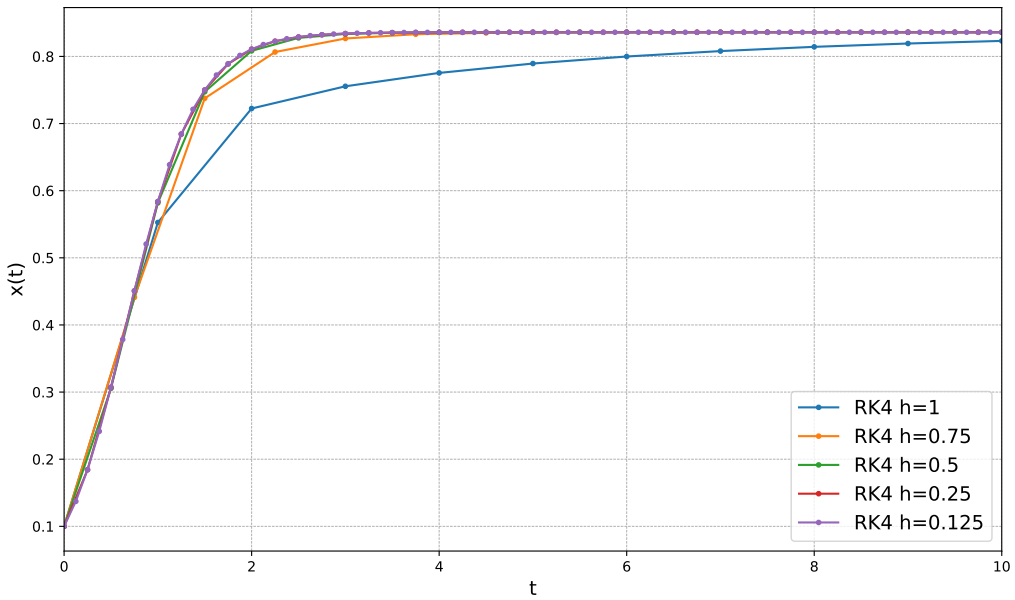

Figura 31: Comparação entre as soluções obtidas pelo MRK de ordem 4

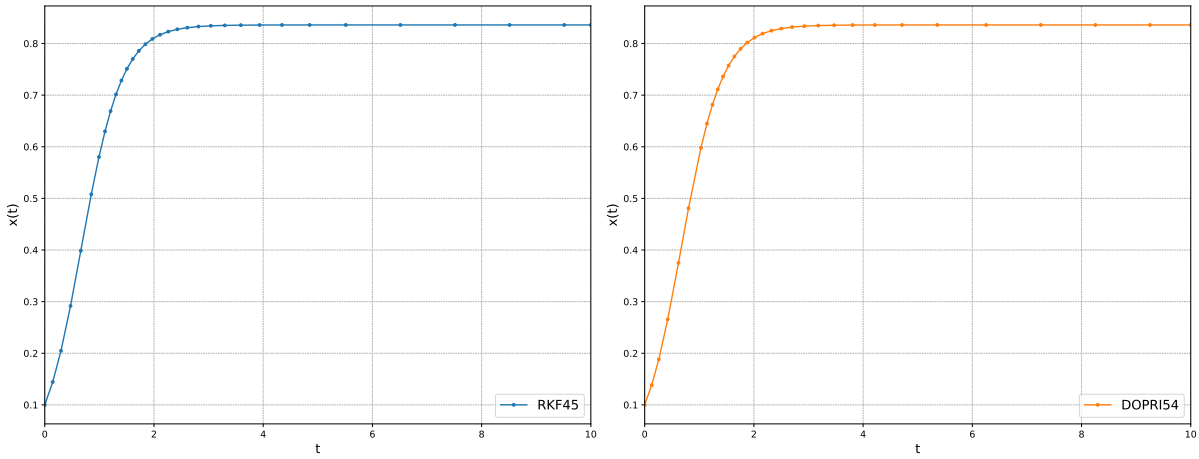

Figura 32: Comparação entre as soluções obtidas pelo MRK adaptativos

Nas tabelas a seguir são apresentados os resultados obtidos por cada método em pontos fixados.

Tabela 162: Comparação dos métodos para h = 1 e r = 3 nos tempos desejados.

t	Euler	RK2	RK4	RKF45	DOPRI54
1.500000000	-	-	-	0.750056032	0.750055910
3.000000000	0.647453957	0.450513102	0.755488633	0.834047470	0.834047105
6.000000000	0.986175156	0.445445152	0.799869991	0.835972144	0.835971936
9.000000000	0.343905338	0.445815136	0.819231346	0.835972854	0.835972842

Tabela 163: Comparação dos métodos para h = 0.75 e r = 3 nos tempos desejados.

t	Euler	RK2	RK4	RKF45	DOPRI54
1.500000000	0.703015220	0.617748177	0.737516774	0.750056032	0.750055910
3.000000000	0.735649323	0.692654750	0.826720000	0.834047470	0.834047105
6.000000000	0.768006320	0.746513611	0.835879699	0.835972144	0.835971936
9.000000000	0.785957175	0.771980906	0.835971915	0.835972854	0.835972842

Tabela 164: Comparação dos métodos para h = 0.5 e r = 3 nos tempos desejados.

t	Euler	RK2	RK4	RKF45	DOPRI54
1.500000000	0.753654554	0.695195945	0.747355622	0.750056032	0.750055910
3.000000000	0.837367079	0.815221071	0.833471838	0.834047470	0.834047105
6.000000000	0.835973827	0.835440045	0.835971090	0.835972144	0.835971936
9.000000000	0.835972854	0.835959007	0.835972852	0.835972854	0.835972842

Tabela 165: Comparação dos métodos para h = 0.25 e r = 3 nos tempos desejados.

t	Euler	RK2	RK4	RKF45	DOPRI54
1.500000000	0.756480859	0.738269229	0.749901667	0.750056032	0.750055910
3.000000000	0.835782293	0.832685892	0.834026734	0.834047470	0.834047105
6.000000000	0.835972853	0.835969606	0.835972027	0.835972144	0.835971936
9.000000000	0.835972854	0.835972850	0.835972853	0.835972854	0.835972842

Tabela 166: Comparação dos métodos para h = 0.125 e r = 3 nos tempos desejados.

t	Euler	RK2	RK4	RKF45	DOPRI54
1.500000000	0.753254098	0.747296672	0.750046428	0.750056032	0.750055910
3.000000000	0.835116176	0.833812822	0.834046227	0.834047470	0.834047105
6.000000000	0.835972784	0.835971783	0.835972050	0.835972144	0.835971936
9.000000000	0.835972854	0.835972853	0.835972853	0.835972854	0.835972842

A análise numérica para o caso com taxa de crescimento $r = 6$ é apresentada a seguir. As Figuras 33, 34, 35 e 36 ilustram a convergência das soluções para os métodos MEE, RK2, RK4 e adaptativos, respectivamente, à medida que o passo de simulação h é refinado.

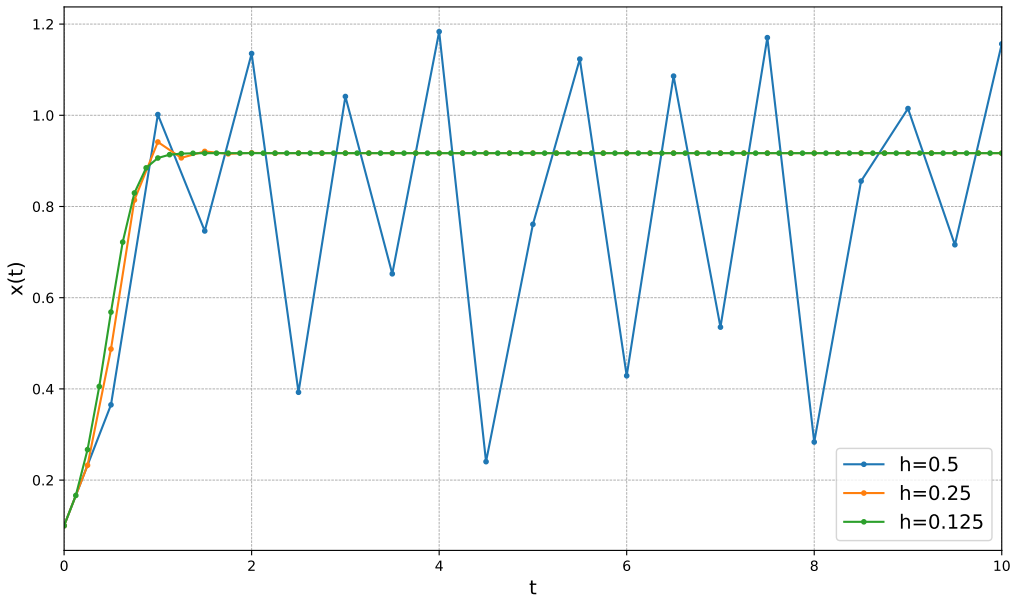

Figura 33: Comparação entre as soluções obtidas pelo MEE

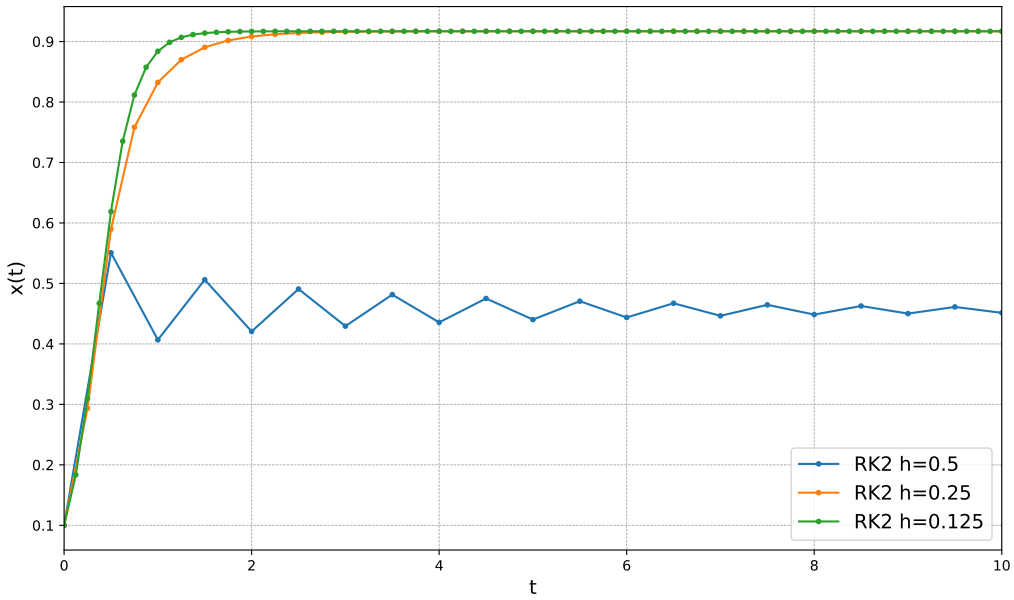

Figura 34: Comparação entre as soluções obtidas pelo MRK de ordem 2

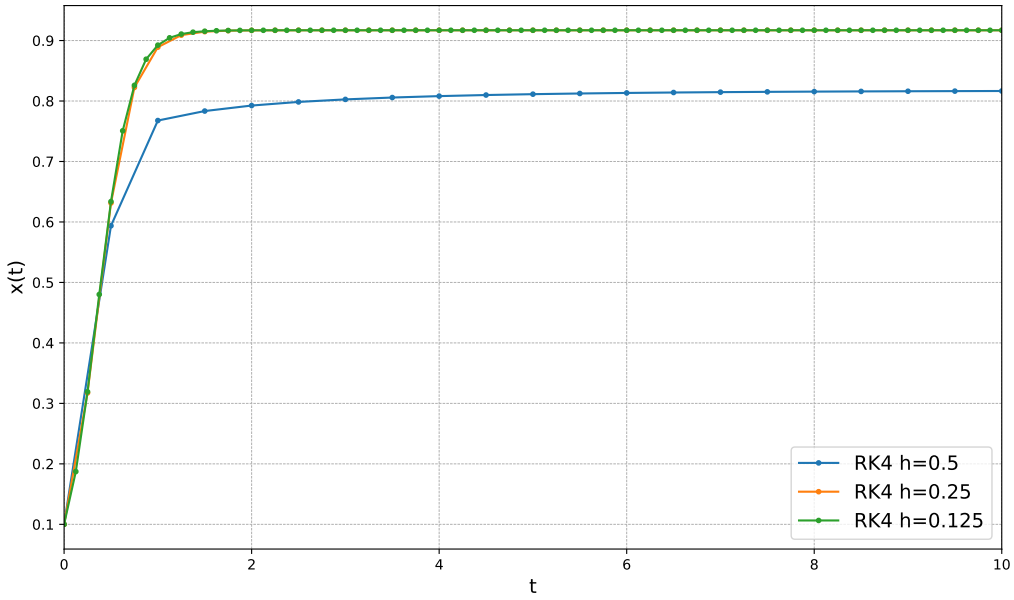

Figura 35: Comparação entre as soluções obtidas pelo MRK de ordem 4

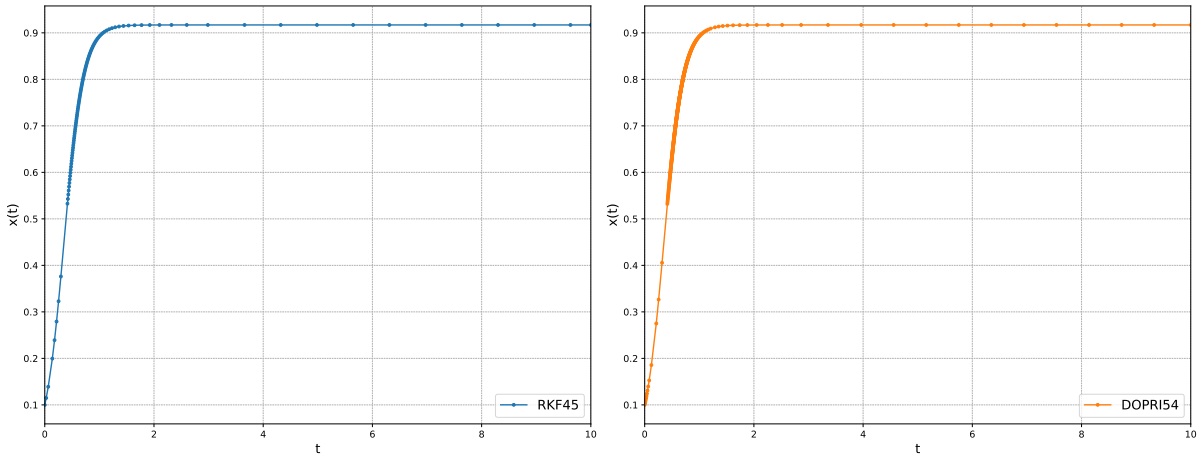

Figura 36: Comparação entre as soluções obtidas pelo MRK adaptativos

Nas tabelas a seguir são apresentados os resultados obtidos por cada método em pontos fixados.

Tabela 167: Comparação dos métodos para h = 0.5 e r = 6 nos tempos desejados.

t	Euler	RK2	RK4	RKF45	DOPRI54
1.500000000	0.746349327	0.506118342	0.783417776	0.915413465	0.915413129
3.000000000	1.041219974	0.429405016	0.802676838	0.916978481	0.916977962
6.000000000	0.428787467	0.443714364	0.813350037	0.916978661	0.916978547
9.000000000	1.014791915	0.450137012	0.816151625	0.916978981	0.916978564

Tabela 168: Comparação dos métodos para h = 0.25 e r = 6 nos tempos desejados.

t	Euler	RK2	RK4	RKF45	DOPRI54
1.500000000	0.920810382	0.890427695	0.914689016	0.915413465	0.915413129
3.000000000	0.916991534	0.916041364	0.916977470	0.916978481	0.916977962
6.000000000	0.916978685	0.916977472	0.916978685	0.916978661	0.916978547
9.000000000	0.916978685	0.916978683	0.916978685	0.916978981	0.916978564

Tabela 169: Comparação dos métodos para h = 0.125 e r = 6 nos tempos desejados.

t	Euler	RK2	RK4	RKF45	DOPRI54
1.500000000	0.916880682	0.913960417	0.915389516	0.915413465	0.915413129
3.000000000	0.916978685	0.916976511	0.916978285	0.916978481	0.916977962
6.000000000	0.916978685	0.916978685	0.916978685	0.916978661	0.916978547
9.000000000	0.916978685	0.916978685	0.916978685	0.916978981	0.916978564

Observa-se que, com o aumento do parâmetro r de 2 para 3 e depois para 6, os métodos de passo fixo, como o MEE e os de Runge-Kutta, apresentam instabilidade ao utilizar passos maiores, como $h = 1$ e $h = 0.75$. Este comportamento, decorrente da inclinação

da curva da solução que se torna mais acentuada com o aumento de r, é contornado pelos métodos de passo adaptativo. O passo h é ajustado, nestes métodos, para manter o erro estimado abaixo de uma tolerância pré-definida, que neste estudo foi de 10^{-6}.

Como não há solução exata, o controle do erro é feito com pares embutidos. Os métodos escolhidos foram o Runge-Kutta-Fehlberg (RKF45) e o Dormand-Prince (DOPRI54). A tabela a seguir apresenta o custo computacional dos métodos adaptativos para $r = 2$, $r = 3$ e $r = 6$. Nela, o **Nº de Pontos** representa os passos bem-sucedidos, enquanto o **Nº de Iterações** corresponde ao total de avaliações da função $f(t, x)$, incluindo os passos que foram rejeitados. A adaptação do passo h foi realizada utilizando a fórmula proposta por Kincaid e Cheney dada por $h_{\text{novo}} = \sigma h_{\text{antigo}} \left(\frac{\delta}{err}\right)^{1/5}$, onde $\sigma = 0.9$, $\delta = 10^{-6}$ é a tolerância e err é o erro local estimado.

Tabela 170: Comparativo de performance entre os métodos de passo adaptativo para diferentes valores do parâmetro r.

Método	**Nº de Pontos**	**Nº de Iterações**
r = 2		
RKF45	25	157
DOPRI54	21	133
r = 3		
RKF45	32	194
DOPRI54	31	220
r = 6		
RKF45	114	792
DOPRI54	476	3 801

Códigos Implementados

Esta seção apresenta os códigos em Python que implementam os diversos métodos numéricos estudados. São exibidas as funções para a resolução de Problemas de Valor Inicial (PVI) com métodos de passo simples (Euler, Série de Taylor, Runge-Kutta), passo adaptativo (Runge-Kutta-Fehlberg, Dormand-Prince) e múltiplos passos (Adams-Bashforth). O código-fonte completo está disponível no repositório do projeto no GitHub (link).

Código 1: Implementação, em Python, da função que calcula o erro relativo.

```
def erro_relativo(y_exato, y_aproximado):
    return np.abs((y_exato - y_aproximado) / y_exato)
```

Código 2: Implementação, em Python, da função que calcula o erro local (absoluto).

```
def erro_local(y_exato, y_aproximado):
    return np.abs(y_exato - y_aproximado)
```

Código 3: Implementação, em Python, do Método de Euler Explícito.

```
def metodo_euler_explicito(func, condicao_inicial, T, dt):
    t0 = condicao_inicial[0]
    y0 = condicao_inicial[1]

    n = int((T - t0) / dt)

    t_out = np.zeros(n + 1)
    y_out = np.zeros(n + 1)

    # Define o primeiro ponto da solucao (a condicao inicial)
    t_out[0] = t0
    y_out[0] = y0

    for i in range(n):
        # Pega os valores do passo anterior (usando o indice i)
        ti = t_out[i]
        yi = y_out[i]

        # A formula do metodo de Euler explicito
        y_out[i + 1] = yi + dt*func(ti, yi)

        # Atualiza o vetor de tempo
        t_out[i + 1] = ti + dt

    return t_out, y_out
```

Código 4: Implementação, em Python, do Método da Série de Taylor.

```
def metodo_serie_taylor(func, condicao_inicial, T, dt):
    t0 = condicao_inicial[0]
```

```
    y0 = condicao_inicial[1]

    n = int((T - t0) / dt)
    ordem = len(func)

    t_out = np.zeros(n + 1)
    y_out = np.zeros(n + 1)

    # Define o primeiro ponto da solucao (a condicao inicial)
    t_out[0] = t0
    y_out[0] = y0

    for i in range(n):
        # Pega os valores do passo anterior (usando o indice i)
        ti = t_out[i]
        yi = y_out[i]

        soma_taylor = 0.0

        for k in range(ordem):
            # Termo (k+1) da serie de Taylor
            valor_derivada = func[k](ti, yi)
            fatorial = math.factorial(k + 1)
            soma_taylor += (dt**(k + 1) / fatorial) * valor_derivada

        y_out[i + 1] = yi + soma_taylor

        # Atualiza o vetor de tempo
        t_out[i + 1] = ti + dt

    return t_out, y_out
```

Código 5: Implementação, em Python, do Método de Runge-Kutta de Ordem 2.

```
def metodo_rk2(func, condicao_inicial, T, dt):
    t0 = condicao_inicial[0]
    y0 = condicao_inicial[1]

    n = int((T - t0) / dt)

    t_out = np.zeros(n + 1)
    y_out = np.zeros(n + 1)

    # Define o primeiro ponto da solucao (a condicao inicial)
    t_out[0] = t0
    y_out[0] = y0

    for i in range(n):
        # Pega os valores do passo anterior (usando o indice i)
        ti = t_out[i]
        yi = y_out[i]
```

```
        f1 = dt*func(ti, yi)
        f2 = dt*func(ti + dt, yi + f1)

        # Formula do metodo de Runge-Kutta de ordem 2
        y_out[i + 1] = yi + (1 / 2) * (f1 + f2)

        # Atualiza o vetor de tempo
        t_out[i + 1] = ti + dt

    return t_out, y_out
```

Código 6: Implementação, em Python, do Método de Runge-Kutta de Ordem 3.

```
def metodo_rk3(func, condicao_inicial, T, dt):
    t0 = condicao_inicial[0]
    y0 = condicao_inicial[1]

    n = int((T - t0) / dt)

    t_out = np.zeros(n + 1)
    y_out = np.zeros(n + 1)

    # Define o primeiro ponto da solucao (a condicao inicial)
    t_out[0] = t0
    y_out[0] = y0

    A = np.array([
        [0,   0,   0],
        [0.5, 0,   0],
        [-1,  2,   0]
    ])
    B = np.array([1/6, 4/6, 1/6])
    C = np.array([0, 0.5, 1])

    for i in range(n):
        # Pega os valores do passo anterior (usando o indice i)
        ti = t_out[i]
        yi = y_out[i]

        f = np.zeros(len(C))
        for j in range(len(C)):
            tj = ti + C[j]*dt
            yj = yi + np.dot(A[j, :j], f[:j])
            f[j] = dt * func(tj, yj)

        y_out[i + 1] = yi + np.dot(B, f)

        # Atualiza o vetor de tempo
        t_out[i + 1] = ti + dt
```

```
    return t_out, y_out
```

Código 7: Implementação, em Python, do Método de Runge-Kutta de Ordem 2 (Modificado).

```
def metodo_rk2_modificado(func, condicao_inicial, T, dt):
    t0 = condicao_inicial[0]
    y0 = condicao_inicial[1]
    alpha = .5
    beta = .5
    w2 = 1 / (2 * alpha)
    w1 = 1 - w2
    n = int((T - t0) / dt)

    t_out = np.zeros(n + 1)
    y_out = np.zeros(n + 1)

    # Define o primeiro ponto da solucao (a condicao inicial)
    t_out[0] = t0
    y_out[0] = y0

    for i in range(n):
        # Pega os valores do passo anterior (usando o indice i)
        ti = t_out[i]
        yi = y_out[i]

        # Calculo dos k1 e k2
        f1 = dt*func(ti, yi)
        f2 = dt*func(ti + dt*alpha, yi + f1*beta)

        # Formula do metodo de Runge-Kutta de ordem 2 modificado
        y_out[i + 1] = yi + w1*f1 + w2*f2

        # Atualiza o vetor de tempo
        t_out[i + 1] = ti + dt

    return t_out, y_out
```

Código 8: Implementação, em Python, do Método de Runge-Kutta de Ordem 4.

```
def metodo_rk4(func, condicao_inicial, T, dt):
    t0 = condicao_inicial[0]
    y0 = condicao_inicial[1]

    n = int((T - t0) / dt)

    t_out = np.zeros(n + 1)
    y_out = np.zeros(n + 1)

    # Define o primeiro ponto da solucao (a condicao inicial)
```

```
    t_out[0] = t0
    y_out[0] = y0

    A = np.array([
        [0,   0,   0, 0],
        [0.5, 0,   0, 0],
        [0,   0.5, 0, 0],
        [0,   0,   1, 0]
    ])
    B = np.array([1/6, 1/3, 1/3, 1/6])
    C = np.array([0, 0.5, 0.5, 1])

    for i in range(n):
        # Pega os valores do passo anterior (usando o indice i)
        ti = t_out[i]
        yi = y_out[i]

        f1 = dt * func(ti, yi)
        f2 = dt * func(ti + C[1]*dt, yi + A[1,0]*f1)
        f3 = dt * func(ti + C[2]*dt, yi + A[2,1]*f2)
        f4 = dt * func(ti + C[3]*dt, yi + A[3,2]*f3)

        y_out[i + 1] = yi + B[0]*f1 + B[1]*f2 + B[2]*f3 + B[3]*f4
        t_out[i + 1] = ti + dt

    return t_out, y_out
```

Código 9: Implementação, em Python, do Método de Runge-Kutta de Ordem 6.

```
def metodo_rk6(func, condicao_inicial, T, dt):
    t0 = condicao_inicial[0]
    y0 = condicao_inicial[1]

    n = int(np.ceil((T - t0) / dt))

    t_out = np.zeros(n + 1)
    y_out = np.zeros(n + 1)

    # Define o primeiro ponto da solucao (a condicao inicial)
    t_out[0] = t0
    y_out[0] = y0

    A = np.array([
        [0, 0, 0, 0, 0, 0, 0],
        [1/3, 0, 0, 0, 0, 0, 0],
        [0, 2/3, 0, 0, 0, 0, 0],
        [1/12, 1/3, -1/12, 0, 0, 0, 0],
        [-1/16, 9/8, -3/16, -3/8, 0, 0, 0],
        [0, 9/8, -3/8, -3/4, 1/2, 0, 0],
        [9/44, -9/11, 63/44, 18/11, 0, -16/11, 0]
    ])
```

```
    B = np.array([11/120, 0, 27/40, 27/40, -4/15, -4/15, 11/120])
    C = np.array([0.0, 1/3, 2/3, 1/3, 1/2, 1/2, 1])

    for i in range(n):
        # Pega os valores do passo anterior (usando o indice i)
        ti = t_out[i]
        yi = y_out[i]

        f = np.zeros(len(C))
        for j in range(len(C)):
            tj = ti + C[j]*dt
            yj = yi + np.dot(A[j, :j], f[:j])
            f[j] = dt * func(tj, yj)

        y_out[i + 1] = yi + np.dot(B, f)

        # Atualiza o vetor de tempo
        t_out[i + 1] = ti + dt

    return t_out, y_out
```

Código 10: Implementação, em Python, dos Métodos de Runge-Kutta Adaptativos (RKF45 e DOPRI54).

```
def metodo_rk(func, condicao_inicial, T, dt0, tol=1e-6, metodo='RKF45'):
    t0 = condicao_inicial[0]
    y0 = condicao_inicial[1]
    n = 0
    ite = 0

    t_out = [t0]
    y_out = [y0]

    t = t0
    y = y0

    match metodo:
        case 'RKF45':
            A = np.array([
                [0,          0,          0,          0,         0,        0],
                [1/4,        0,          0,          0,         0,        0],
                [3/32,       9/32,       0,          0,         0,        0],
                [1932/2197, -7200/2197,  7296/2197,  0,         0,        0],
                [439/216,   -8,          3680/513,   -845/4104,  0,        0],
                [-8/27,      2,         -3544/2565,  1859/4104, -11/40,   0]
            ])
            B4 = np.array([25/216, 0, 1408/2565, 2197/4104, -1/5, 0])
            B5 = np.array([16/135, 0, 6656/12825, 28561/56430, -9/50, 2/55])
            C = np.array([0, 1/4, 3/8, 12/13, 1, 1/2])

        case 'DOPRI54':
```

```
        A = np.array([
            [0, 0, 0, 0, 0, 0, 0],
            [1/5, 0, 0, 0, 0, 0, 0],
            [3/40, 9/40, 0, 0, 0, 0, 0],
            [44/45, -56/15, 32/9, 0, 0, 0, 0],
            [19372/6561, -25360/2187, 64448/6561, -212/729, 0, 0, 0],
            [9017/3168, -355/33, 46732/5247, 49/176, -5103/18656, 0, 0],
            [35/384, 0, 500/1113, 125/192, -2187/6784, 11/84, 0]
        ])
        B5 = np.array([35/384, 0, 500/1113, 125/192, -2187/6784, 11/84, 0])
        B4 = np.array([5179/57600, 0, 7571/16695, 393/640, -92097/339200,
            187/2100, 1/40])
        C = np.array([0, 1/5, 3/10, 4/5, 8/9, 1, 1])


    while t < T:
        dt = dt0
        if t + dt > T:
            dt = T - t

        f = np.zeros(len(C))
        for i in range(len(C)):
            ti = t + C[i]*dt
            yi = y + np.dot(A[i, :i], f[:i])
            f[i] = dt * func(ti, yi)

        y4 = y + np.dot(B4,f)
        y5 = y + np.dot(B5, f)
        erro = np.abs(y5 - y4)

        # Controle adaptativo do passo
        i = 0
        while erro >= tol and i < 10:
            dt = .9 * dt * (tol / erro)**(1/5) # Kincaid e Cheney
            if t + dt > T:
                dt = T - t

            for i in range(len(C)):
                ti = t + C[i]*dt
                yi = y + np.dot(A[i, :i], f[:i])
                f[i] = dt * func(ti, yi)

            y4 = y + np.dot(B4,f)
            y5 = y + np.dot(B5, f)
            erro = np.abs(y5 - y4)
            i += 1

        if i == 10 and erro >= tol:
            print("Aviso: numero maximo de iteracoes internas atingido, erro
                ainda acima da tolerancia!")
```

```
        t += dt
        y = y5
        t_out.append(t)
        y_out.append(y)
        n += 1
        ite += 1 + i

    return np.array(t_out), np.array(y_out), n, ite
```

Código 11: Implementação, em Python, do Método de Adams-Bashforth de Ordem 2.

```
def metodo_ab2(func, condicao_inicial, T, dt):
    t0 = condicao_inicial[0]
    y0 = condicao_inicial[1]

    n = int((T - t0) / dt)

    t_out = np.zeros(n + 1)
    y_out = np.zeros(n + 1)

    # Define o primeiro ponto da solucao (a condicao inicial)
    t_out[0] = t0
    y_out[0] = y0

    # Primeiro passo usando rk4
    t_out[1] = t_out[0] + dt
    y_out[1] = metodo_rk4(func, [t_out[0], y_out[0]], t_out[1], dt)[1][1]

    for i in range(1, n):
        ti = t_out[i]
        yi = y_out[i]
        t_ant = t_out[i - 1]
        y_ant = y_out[i - 1]

        # Formula do metodo de Adams-Bashforth de ordem 2
        y_out[i + 1] = yi + (dt / 2) * (3*func(ti, yi) - func(t_ant, y_ant))

        # Atualiza o vetor de tempo
        t_out[i + 1] = ti + dt

    return t_out, y_out
```

Código 12: Implementação, em Python, do Método de Adams-Bashforth de Ordem 3.

```
def metodo_ab3(func, condicao_inicial, T, dt):
    t0 = condicao_inicial[0]
    y0 = condicao_inicial[1]

    n = int((T - t0) / dt)
```

```
    t_out = np.zeros(n + 1)
    y_out = np.zeros(n + 1)

    # Define o primeiro ponto da solucao (a condicao inicial)
    t_out[0] = t0
    y_out[0] = y0

    # Primeiro passo usando RK4
    t_out[1] = t_out[0] + dt
    y_out[1] = metodo_rk6(func, [t_out[0], y_out[0]], t_out[1], dt)[1][1]

    # Segundo passo usando RK4
    t_out[2] = t_out[1] + dt
    y_out[2] = metodo_rk6(func, [t_out[1], y_out[1]], t_out[2], dt)[1][1]

    for i in range(2, n):
        ti = t_out[i]
        yi = y_out[i]
        t_ant1 = t_out[i - 1]
        y_ant1 = y_out[i - 1]
        t_ant2 = t_out[i - 2]
        y_ant2 = y_out[i - 2]

        # Formula do metodo de Adams-Bashforth de ordem 3
        y_out[i + 1] = yi + (dt / 12) * (23*func(ti, yi) - 16*func(t_ant1,
            y_ant1) + 5*func(t_ant2, y_ant2))

        # Atualiza o vetor de tempo
        t_out[i + 1] = ti + dt

    return t_out, y_out
```

Código 13: Implementação, em Python, do Método de Adams-Bashforth de Ordem 4.

```
def metodo_ab4(func, condicao_inicial, T, dt):
    t0 = condicao_inicial[0]
    y0 = condicao_inicial[1]

    n = int((T - t0) / dt)

    t_out = np.zeros(n + 1)
    y_out = np.zeros(n + 1)

    # Define o primeiro ponto da solucao (a condicao inicial)
    t_out[0] = t0
    y_out[0] = y0

    # Primeiro passo usando RK4
    t_out[1] = t_out[0] + dt
    y_out[1] = metodo_rk6(func, [t_out[0], y_out[0]], t_out[1], dt)[1][1]
```

```
    # Segundo passo usando RK4
    t_out[2] = t_out[1] + dt
    y_out[2] = metodo_rk6(func, [t_out[1], y_out[1]], t_out[2], dt)[1][1]

    # Terceiro passo usando RK4
    t_out[3] = t_out[2] + dt
    y_out[3] = metodo_rk6(func, [t_out[2], y_out[2]], t_out[3], dt)[1][1]

    for i in range(3, n):
        ti = t_out[i]
        yi = y_out[i]
        t_ant1 = t_out[i - 1]
        y_ant1 = y_out[i - 1]
        t_ant2 = t_out[i - 2]
        y_ant2 = y_out[i - 2]
        t_ant3 = t_out[i - 3]
        y_ant3 = y_out[i - 3]

        # Formula do metodo de Adams-Bashforth de ordem 4
        y_out[i + 1] = yi + (dt / 24) * (55*func(ti, yi) - 59*func(t_ant1,
            y_ant1) + 37*func(t_ant2, y_ant2) - 9*func(t_ant3, y_ant3))

        # Atualiza o vetor de tempo
        t_out[i + 1] = ti + dt

    return t_out, y_out
```

Código 14: Implementação, em Python, do Método de Adams-Bashforth de Ordem 6.

```
def metodo_ab6(func, condicao_inicial, T, dt):
    t0 = condicao_inicial[0]
    y0 = condicao_inicial[1]

    n = int((T - t0) / dt)

    t_out = np.zeros(n + 1)
    y_out = np.zeros(n + 1)

    # Define o primeiro ponto da solucao (a condicao inicial)
    t_out[0] = t0
    y_out[0] = y0

    # Primeiro passo usando RK4
    t_out[1] = t_out[0] + dt
    y_out[1] = metodo_rk6(func, [t_out[0], y_out[0]], t_out[1], dt)[1][1]

    # Segundo passo usando RK4
    t_out[2] = t_out[1] + dt
    y_out[2] = metodo_rk6(func, [t_out[1], y_out[1]], t_out[2], dt)[1][1]

    # Terceiro passo usando RK4
```

```
    t_out[3] = t_out[2] + dt
    y_out[3] = metodo_rk6(func, [t_out[2], y_out[2]], t_out[3], dt)[1][1]

    # Quarto passo usando RK4
    t_out[4] = t_out[3] + dt
    y_out[4] = metodo_rk6(func, [t_out[3], y_out[3]], t_out[4], dt)[1][1]

    # Quinto passo usando RK4
    t_out[5] = t_out[4] + dt
    y_out[5] = metodo_rk6(func, [t_out[4], y_out[4]], t_out[5], dt)[1][1]

    for i in range(5, n):
        ti = t_out[i]
        yi = y_out[i]
        t_ant1 = t_out[i - 1]
        y_ant1 = y_out[i - 1]
        t_ant2 = t_out[i - 2]
        y_ant2 = y_out[i - 2]
        t_ant3 = t_out[i - 3]
        y_ant3 = y_out[i - 3]
        t_ant4 = t_out[i - 4]
        y_ant4 = y_out[i - 4]
        t_ant5 = t_out[i - 5]
        y_ant5 = y_out[i - 5]

        # Formula do metodo de Adams-Bashforth de ordem 6
        y_out[i + 1] = yi + (dt / 1440) * (4277*func(ti, yi) - 7923*func(t_ant1,
            y_ant1) + 9982*func(t_ant2, y_ant2) - 7298*func(t_ant3, y_ant3) +
            2877*func(t_ant4, y_ant4) - 475*func(t_ant5, y_ant5))

        # Atualiza o vetor de tempo
        t_out[i + 1] = ti + dt

    return t_out, y_out
```